A Character-Word Compositional Neural Language Model for Finnish

Matti Lankinen
Reaktor
matti.lankinen@reaktor.com

Hannes Heikinheimo
Reaktor
hannes.heikinheimo@reaktor.com

Pyry Takala
Aalto University
pyry.takala@aalto.fi

Tapani Raiko
Aalto University
tapani.raiko@aalto.fi

Juha Karhunen
Aalto University
juha.karhunen@aalto.fi

Abstract

Inspired by recent research, we explore ways to model the highly morphological Finnish language at the level of characters while maintaining the performance of word-level models. We propose a new Character-to-Word-to-Character (C2W2C) compositional language model that uses characters as input and output while still internally processing word level embeddings. Our preliminary experiments, using the Finnish Europarl V7 corpus, indicate that C2W2C can respond well to the challenges of morphologically rich languages such as high out of vocabulary rates, the prediction of novel words, and growing vocabulary size. Notably, the model is able to correctly score inflectional forms that are not present in the training data and sample grammatically and semantically correct Finnish sentences character by character.

1   Introduction

Character level language models have attracted significant research attention recently in the deep learning community [18, 5, 20, 2, 14, 15]. Character level language models have certain advantages over word level models, which tend to struggle with applications that require a large vocabulary or need to tackle high out-of-vocabulary word rates. In many cases increasing vocabulary size quickly multiplies the parameter count, making word-level models complex and slow to train. Furhermore, with highly inflectional languages such as Turkish, Russian, Hungarian or Finnish word level models fail to encode part-of-speech information embedded into the morphology of inflected words.

Among highly morphological languages Finnish is one of the most complex. Hence the Finnish language provids an interesting testbed for research in character level language models. It has a large number of inflectional types for both nouns and verbs and it uses suffixes to express grammatical relations. As an example, the single Finnish word *talossanikin* corresponds to the English phrase *in my house, too*. Bojanowski et al. [2] report the vocabulary size and out-of-vocabulary rate of various European languages for a set of experiments on the Europarl dataset [17]. For Finnish these values were reported to be 10 % and 37 % higher respectively, compared to Hungarian, which had the second hightest reported values.
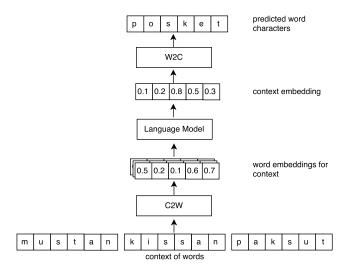
Figure 1: C2W2C model overall structure

In this study we propose a new **Character-to-Word-to-Character** (C2W2C) compositional language model for modeling Finnish. The model combines ideas from [20] and [5], and consists of three logical parts: (1) the Character-to-Word C2W model inspired by [20], (2) a traditional LSTM Language Model [28], and (3) a Word-to-Character W2C decoder model inspired by [5]. The entire model is implemented sequentially, so the output from any logical part can be used as an input for the next one. Thus, the model can be trained and used with a single pass without any preliminary sub-word tokenization or word classification. Also, all the parts are independent, so they can be trained individually. The structure of C2W2C model is described in Figure 1.

The rest of our discussion is organised as follows: Section 2 discusses related work. Section 3 describes our language model architecture in more detail. Section 4 and 5 present our experiment setup and the model performance results. Section 6 contains the conclusions.

2   Related work

The usual approach in neural language modeling is to use so-called one-hot vectors [25] to represent input and output words of the model. However, the number of parameters linearly follow the size of the vocabulary, hence resulting into scalability issues for large vocabularies. Ling et .al [20] proposed a new Character-to-Word model (C2W) to solve this issue. In C2W, the traditional word-level projection layer is replaced with a character-level Bidirectional LSTM unit that produced word embeddings to the language model LSTM. As a result, the input layer parameters are reduced from 4M to 180k with a dataset from the Wikipedia corpus. Also noticeable performance improvements were reported compared to traditional word-level models. Here we use the C2W model of [20] as part of our C2W2C model.

In [15] Kim et al. propose a convolutional neural network and a highway network over characters (CharCNN), whose output is given to a LSTM language model. With this setup, the authors managed in par performance with state-of-art models with 60% fewer parameters. The work in [14] proposes also a solution by using convolution networks, but use a CharCNN as a part of the output layer. The architecture of this model followes the original work of [15], but instead of having the projection layer for output, they use a CharCNN to create word embeddings, which they combine with a context vector from a language model layer to produce an output word logit function. Other alternative approaches in dealing with the large output vocabulary size are the hierachical softmax [11, 23], importance sampling [1], and noice contrast estimation [12, 24]. Although these methods work well and give speedups without significant performance penalties, they do not remove the challenges of out-of-vocabulary words.

0.9 | 0.2 | 0.7 | 0.5 | 0.6   Word embedding **w**

Bi-LSTM

Combine (+)

Forward LSTM   Backwards LSTM

Character embeddings $\mathbf{c}_1^E...\mathbf{c}_m^E$

```
0.3 0.5 0.2 0.8 0.9 0.1
0.2 0.1 0.3 0.4 0.2 0.2
0.4 0.2 0.2 0.4 0.3 0.9
0.7 0.1 0.6 0.7 0.1 0.4
 .   .   .   .   .   .
 .   .   .   .   .   .
0.1 0.6 0.1 0.8 0.6 0.2
```

Character lookup table
(projection **P**)

Index vectors $\mathbf{c}_1^I...\mathbf{c}_m^I$

```
0 0 1 0 0 1
0 0 0 0 1 0
0 1 0 0 0 0
1 0 0 0 0 0
. . . . . .
0 0 0 1 0 0
```
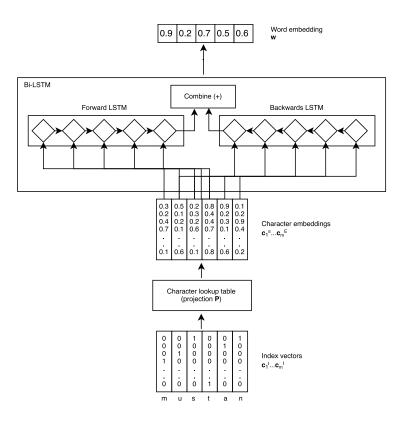
m u s t a n

Figure 2: The structure of C2W model

Cho et al. [5] proposed a new two-stage encoder-decoder RNN performing machine translations from characters to characters. They use the previous encoder-decoder work of [4] and gated-feedback networks [6] as a basis and built an adaptive "Bi-Scale Recurrent Neural Network", where the idea is to model sentences by using two layers. The "faster layer" models the fast-changing timescale (i.e. words), and the "slower layer" modeles the slower-changing timescale (i.e. characters). The layers were gated so that the produced context vector was an adaptive combination of the activations of both layers, and that context was used to decode the translation characters. In this paper we use a character-level decoder as a part of its C2W2C model similar to [5].

Recent research, including [22, 2, 21] have indicated the need for more research on subword level models for morphologically rich languages. However, sub-word level language models have been studies previously. Creutz and Lagus [7] introduced an unsupervised morpheme segmentation and morphology induction algorithm called Morfessor. Vilar et al. [29] translated sentences character by character, but the results of this technique were inferior to those of word-based approaches. Botha and Blunsom [3] proposed a word segment-level language model.

In this paper we apply neural language modelling especilly to the Finnish language. Previous work done towards tackling the complexites of Finnish morpoholgy include [19] and [26]. Most recent work on neural language modeling applied to the Finnish language are [9] and [8]. Chung et al. [5] demonstrate their machine translation work using Finnish as one of their target languages.

3   A compositional language model

3.1   Character to Word model

The character-to-word (C2W) model is the first part of the C2W2C model. The implementation mainly follows [20]. The responsibility of C2W in C2W2C is to transform the input context word sequence $\omega_1...\omega_n$ into d_W-dimensional *word embeddings* $\mathbf{w}_1...\mathbf{w}_n$. The word embeddings capture
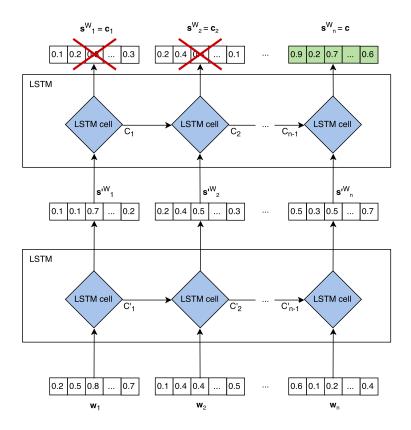
Figure 3: The C2W2C language model structure

the characteristics of each of the words in the given context. These characteristics contain, for example, part-of-speech information (noun, verb, adjective, etc.), grammatical tense, and inflection information that is usually encoded into the words in highly inflectional languages (e.g. Finnish words *auto, autossa, autosta* as opposed to their English counterparts *a car, in the car, from the car*).

C2W2C contains the first two layers from the original C2W model implementation: (1) character lookup table and (2) Bi-LSTM generating the word embeddings for the given word sequence $\omega_1...\omega_n$. These first two C2W layers are illustrated in Figure 2. More details can be found from [20].

3.2 Language Model

The language model component of C2W2C is a standard 2-layer LSTM network [13] that takes word embeddings $\mathbf{w}_1...\mathbf{w}_n$ as an input and yields the state sequence $\mathbf{s}_0^W...\mathbf{s}_n^W$ as an ouput. In training time, the intermediate states $\mathbf{s}_0^W...\mathbf{s}_{n-1}^W$ are discarded and the actual *context embedding* $\mathbf{c}$ is selected by taking the last state of the computed state sequence. The overall structure of the LM of C2W2C is shown in Figure 3.

3.3 Word to Character model

The Word to Character (W2C) model is the final part of C2W2C mode. Its goal is to decompose the context embedding $\mathbf{c}$ back to a sequence of characters along the lines of the work of [5] and [4].

The full implementation of the character-level encoder-decoder [5] contains two adaptive decoder-RNN [4] layers whose goal is to translate characters from a source word sequence $\mathbf{x}_1...\mathbf{x}_n$ into a target word sequence $\mathbf{y}_1...\mathbf{y}_n$. However, since C2W2C tries to predict only a single target word, our W2C implementation has only one layer following the details of the original decoder-RNN.
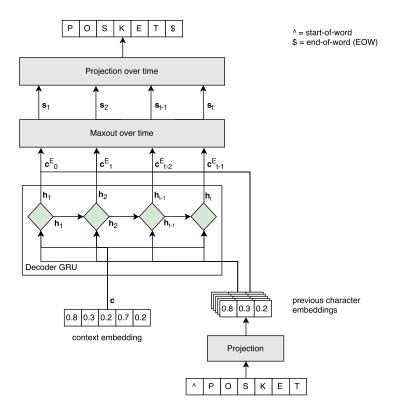
Figure 4: The structure of W2C model

W2C works as follows: The layer receives the context embedding $\mathbf{c}$ from the language model and combines it with the predecessor character c_{t-1} of the predicted character c_i. By using these two components, the RNN decoder maintains a hidden state $\mathbf{h}$ over every predicted character. The used RNN implementation is a variant of Gated Recurrent Unit [4], with the difference that instead of relying only on the hidden state $\mathbf{h}_{t-1}$, the RNN also uses an embedding of the predecessor character $\mathbf{c}_{t-1}^E$ to get the next state $\mathbf{h}_t$ by using the following formula:

$$
\begin{aligned}
\mathbf{h}_t &= z_t \mathbf{h}_{t-1} + (1 - z_t)\mathbf{h}'_t \\
\mathbf{h}'_t &= \tanh(\mathbf{W}_h \mathbf{c}_{t-1}^E + r_t(\mathbf{U}_h \mathbf{h}_{t-1} + \mathbf{C}_h \mathbf{c})) \\
z_t &= \sigma(\mathbf{W}_z \mathbf{c}_{t-1}^E + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{C}_z \mathbf{c}) \\
r_t &= \sigma(\mathbf{W}_r \mathbf{c}_{t-1}^E + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{C}_r \mathbf{c})
\end{aligned}
\tag{1}
$$

The initial $\mathbf{c}_0^E$ is a zero vector, and the initial hidden state is initialized by using the context vector and $\tanh$ activation:

$$
\mathbf{h}_0 = \tanh(\mathbf{V}\mathbf{c})
$$

Each state $\mathbf{h}_t$ of the received state sequence $\mathbf{h}_1 ... \mathbf{h}_m$ is combined with the context vector and the previous character embedding $\mathbf{c}_{t-1}^E$, and the logit for the predicted character index (in V_C) is received by running the combined embedding through a 2-feature Maxout network [10] and projecting the result to the V_C index space:

$$
\begin{aligned}
\mathbf{c}_t^I &= \mathbf{P}_I \mathbf{s}_t \\
\mathbf{s}_t &= \max\{\mathbf{s'}_t^1, \mathbf{s'}_t^2\} \\
\mathbf{s'}_t^i &= \mathbf{O}_h^i \mathbf{h}_t + \mathbf{O}_e^i \mathbf{c}_{t-1}^E + \mathbf{O}_c^i \mathbf{c} + \mathbf{b}
\end{aligned}
\tag{2}
$$

5

Probabilities of the predicted characters are received by running Softmax activation over the logit vectors. The overall structure of W2C is illustrated in Figure 4.

4   Experiment: Finnish Language Modeling

We examine how well the C2W2C model can model the Finnish language. We are interested in the models capability to learn word-level relationships, find the meaning of the different inflection forms (e.g. *auton*, 'of the car', *autossa*, 'in the car') as well as cope with part of speech information (e.g. *Pekka ajaa punaista autoa*, 'Pekka drives a red car') of a word in the context of a sentence.

4.1   Dataset

We chose the text corpus of Finnish translations of Europarl Parallel Corpus V7[1] [17] as our test dataset. The text data was was tokenized with Apache OpenNLP open-source tool[2] and using Finnish tokenizing model from Finnish Dependency Parser project[3]. The tokenized text data was converted into a lower-case form. No further pre-processing or word segmentation was done to the dataset.

The training set was selected by using the first one million tokens from the tokenized lower-case corpus. The properties of the selected dataset reflected the nature of highly inflectional languages; the one million tokens included *88,000* unique tokens which cover 8.8% from the dataset. 5,000 most frequent tokens covered 78.41 % from the dataset, 10k most frequent tokens 84.97 % and 20k most common frequent tokens 90.55 %.
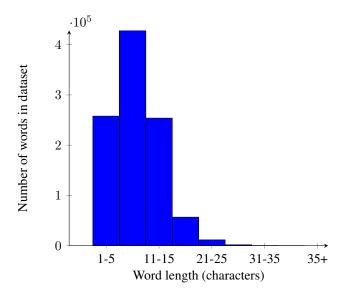

Figure 5: Number of words in dataset by word length

In addition to the training data, extra 10k tokens were fetched from the same Europarl Finnish corpus for test and validation purposes. Both training and validation datasets were pre-processed as described above and sentences were split per line and shuffled. The training dataset sentences were also shuffled at the beginning of each epoch.

---

[1]http://www.statmt.org/europarl
[2]https://opennlp.apache.org
[3]https://github.com/TurkuNLP/Finnish-dep-parser

4.2 Hyper-parameters

The hyper-parameters of C2W model were set to follow the values from the original C2W proposed by [20]. The projection layer $\mathbf{P}_E \in \mathbb{R}^{d_C \times |V_C|}$ for the character embeddings was set to use character properties $d_C = 50$. The intermediate Bi-LSTM cell states $\mathbf{s}_m^f$ and $\mathbf{s}_0^b$ were both set to use $d_{WI} = |\mathbf{s}_m^f| = |\mathbf{s}_0^b| = 150$. Word features number d_W was set to 50. The language model LSTM hidden state d_L was set to 500 for both nodes. W2C decoder's hidden state was set to $|\mathbf{h}| = 500$ and character embeddings for both c_{t-1}^E and $\mathbf{s}_t$ were set to use d_C.

Another variable that has a great influence on the model speed is the maximum length of the input and output words: the longer the maximum length, the more inner LSTM steps must be performed to construct the word embeddings. Fig. 5 shows the word frequency distribution by word length. We chose the maximum word length to be 20, which covers over 98% from the training dataset.

The training was performed by using Backpropagation through time with unlimited context size and Adam optimizer [16] with learning rate of 0.0001and norm clipping with value 2.0. Gradient updates were performed after each time step, and language model's LSTM states were reset before each epoch. In order to speed up the calculation, the data was arranged into batches so that each sample x_i^t in the batch was a predecessor of the next sample x_i^{t+1}. Size of these mini-batches were set to 150. To prevent overfitting we used dropout layers [27] after each submodel (W2C, LM, W2C) with friction value 0.5.

4.3 Perplexity

For character-level perplexity the probability of a word is seen as a product of the probabilities of its characters. Because C2W2C already produces categorical probability distributions by using softmax as its output layer activation, the perplexity for C2W2C was calculated by using the received probability distribution and getting the loss character by character. Those character losses are then added per word, and finally, the word-level cross-entropies are combined and normalised with the testing set size to get overall validation perplexity.

4.4 Environment

The model was trained and validated by using a single Ubuntu 14.04 server having 8GB RAM, single Intel i5 for the model building, and one nVidia GeForce GTX 970 GPU Unit with 3.5GB graphics memory and CUDA (cMEM and cuDNN disabled) for the model training.

The model was implemented by using Python 2.7, Theano 0.8.2[4] and Keras 1.0.4[5]. The source codes of the final C2W2C model implementation can be found at GitHub: `https://github.com/milankinen/c2w2c` under MIT license.

5 Results and Analysis

The model was run by using the dataset, hyperparameters, and environment described in the previous section. In addition to C2W2C, a traditional word-level model ("Word-LSTM" onwards) was trained for comparison. Word-LSTM had the same two-level LSTM language model as C2W2C (as described in subsection 3.2) and typical feedforward projection layers for both inputs and outputs. For the sake of efficiency, context vector $\mathbf{c}|_{dim(\mathbf{c})=500}$ was projected into 150 dimensional space before output projection (removing 31M parameters from the model).

5.1 Quantitative analysis

5.1.1 Model Complexity and training times

As shown in Table 1 the C2W2C model operates in our experiment by using only 25% of the traditional word-level model parameters. The C2W layer, in particular, seems very useful as it reduces the input parameters by a factor of 20 without affecting model performance negatively.

---

[4]http://deeplearning.net/software/theano
[5]http://keras.io

C2W2C			Word-LSTM		
C2W	**LM**	**W2C**	**FF-NN**	**LM**	**FF-NN**
0.26M	3.1M	2.04M	4.5M	3.1M	13.2M
5.41M			20.8M		

Table 1: Model parameters per sub-model

However, even if there are significantly fewer parameters, the training times are roughly equal: Both models can process approximately 900 words per second, resulting in 20 minutes per epoch with the given training data set. However, for the C2W2C the gradient is less stable compared to the World-LSTM model. To remedy this we had to resort to a smaller learning rate, hence requiring more iterations for similar results.

5.1.2 Perplexity

When measured with perplexity, the classic word-level model surpasses C2W2C. With the given dataset, word-level model achieved PP 392.28, whereas PP for C2W2C was 410.95. One possible factor that raises the PP of C2W2C is that it gives predictions to *all* possible character sequence, which means $|V_C|^{maxlen}$ different combinations. However, the predicted vocabulary size is much smaller. The work done in [14] eliminated this by doing an expensive normalisation over the training vocabulary, which indeed improved character-level performance a little bit. However, such normalisation was not done in the scope of this study.

5.2 Qualitative analysis

5.2.1 Grammar

The following table displays some example scores (a length-normalized log-loss of the sentence, where a lower score is better) of sentences with different properties. The bolded word **w** in the first column represents the examined word and the second column lists the tested variants for **w**. None of the listed sentences is found from the training dataset.

The results indicate that C2W2C can learn the stucture of Finnish grammar and capture morphological information that is encoded into words. Especially the sense of the word (e.g. *esittää, esitti, on esittänyt*'to present, presented, has presented') is a property where correct words yield better scores than incorrect ones. Same applies to inflection forms (e.g. väittei*stä*, väittei*ltä*, väittei*lle*, 'from the arguments', 'of the arguments', 'for the arguments') especially when there are more than two subsequent words having same inflection form.

However, C2W2C is less good at predicting dual forms (whether a word is in singular or plural form) — usually the model prefers singular forms. One possible reason for this might be the score function; word score in C2W2C is not length-normalized, which prefers shorter singular words (e.g. *talo* versus *talot* 'house' vesus 'houses', *äänestyksen* versus *äänestyksien* 'vote' versus 'votes').

Perhaps the most interesting feature of the C2W2C model is how well it can predict words that are not part of the dataset. Whereas traditional word-level models would use an unknown token for such words, C2W2C can give scores to the actual words as long as they do not contain any unknown characters. For example, even if words *sijoitusrahasto* 'mutual Fund' and *monivuotisella* 'multiannual' in Table 2 do not belong to the training data, C2W2C can still assign a better scores to the grammatical correct unseen wordform than to their morphological variants found from the dataset!

5.2.2 Semantics

Results indicate that C2W2C can learn semantic meanings and part-of-speech information — for example, if a verb is replaced with a noun, it usually yields worse scores than the correct sentence. The same applies to semantics: for example, entities that can do something usually yield better scores when placed before verbs than passive entities in the same position. However, punctuation seems to be harder for C2W2C. Probably due to a significant number of statements and the nature

Sentence	Variants	Score
<S> Olen huolestunut esitetyistä **w** . </S>	väitteistä*	**5.143**
	väite	6.315
<S> Komission on **w** aloitteen . </S>	esittänyt*	**2.972**
	esittää	4.254
<S> Nämä esitykset vaikuttavat **w** . </S>	järkeviltä*	5.557
	järkevältä	**5.045**
<S> Tämä **w** vaikuttaa kannattavalta . </S>	sijoitusrahasto*	**6.503**
	sijoitusrahastot	7.605
<S> Säästöt aiotaan saavuttaa tällä **w** ohjelmalla . </S>	monivuotisella*	**6.503**
	monivuotista	7.658
	monivuotinen	7.605

Table 2: Word variant scores (lower is better) with different inflection forms. The grammatically correct alternative marked with *.

of the dataset the model typically assigns better scores to periods than question marks, even if the sentence starts with a question word.

A few example sentences and their scores are shown in Table 3. The notation and used scoring are same as in Table 2.

5.3 Sampling text with C2W2C

As an example application, the trained C2W2C model was configured to generate the Finnish political text character by character. Two different strategies were tested: stochastic sampling and beam search sampling. Stochastic sampling is a simple approach where given a context of words, the next most likely word is chosen and used as a part of the context when predicting the next character. Beam search is a heuristic search algorithm that holds k most likely word sequences and expands them until all sequences are terminated (</S> is encountered).

Within those two strategies, beam search gave significantly better results, whereas stochastic sampling usually ended up looping certain phrases. Beam search also had some phrases and words it preferred over others, but the generated samples were much more natural, and the inflection of words was better than with stochastic sampling. A two-layer search beam was used for the text sampling. Individual words were sampled with the beam of $k = 20$, and the sampled words and their probabilities were used with the sentence-level beam of $k = 10$.

The C2W2C model can produce, grammatically and semantically, sensible text, character by character. Some of the best example sentences are displayed below (capitalization was added, and extra spaces were removed afterwards). The first two words of each sentence were given as an initial context, and the rest of the words were sampled by the model.

<S> Haluan kiittää sydämellisesti puheenjohtajavaltio Portugalia hänen mietinnöstään. [6] </S>

<S> Nämä ovat erittäin tärkeitä eurooppalaisia mahdollisuuksia Euroopan Unionin jäsenvaltioiden perustamissopimuksen yhteydessä.[7] </S>

<S> Siitä huolimatta haluaisin onnitella esittelijää hänen erinomaisesta mietinnöstään. [8]</S>

<S> Tämä merkitsee sitä, että Euroopan Unionin perustamissopimuksen voimaantulon tarkoituksena on kuitenkin välttämätöntä eurooppalaisille jäsenvaltioille.[9] </S>

---

[6]I would like to cordially thank the Portuguese Presidency for his report.

[7]These are very important European opportunities in the context of the Treaty of the European Union Member States.

[8]Nevertheless I would like to congratulate the rapporteur on his excellent report.

[9]This means that the entry into force of the Treaty of the European Union's aim is, however, indispensable for the European Member States.

Sentence	Variants	Score
<S> **w** on pyytänyt uutta äänestystä . </S>	Belgia*	**5.160**
	elokuu	6.963
	ilmiö	5.345
<S> Parlamentti **w** yksimielisesti asetuksesta . </S>	päätti*	**5.095**
	komissio	7.339
	suuri	6.628
<S> Oletteko tietoisia , että uhka on ohi **w** </S>	?*	5.455
	.	**5.164**
	,	6.154

Table 3: Sentence scores (lower is better) with different semantical variants. The grammatically correct alternative marked with *.

6 Conclusions

We propose a new C2W2C character-to-word-to-character compositional language model and estimate its performance compared to the traditional word-level LSTM model. The experiments and their validations are done using Finnish language and the Europarl V7 corpus.

Our experiments are sill preliminary, however, the discoveries mainly follow the findings of [14] and [5] — it is indeed possible to model succesfully a morphologically rich language such as Finnish character by character and to significantly reduce model parameters, especially with large corpuses. Larger scale experiments still need to be conducted, including other data sets and languages, to make further conclusions from the results.

With our setup the traditional word-level model still slightly outperforms the C2W2C model in terms of perplexity and convergence speed, but the C2W2C model compensates by successfully handling out-of-vocabulary words. The model can also learn the Finnish grammar and assign better scores to grammatically correct sentences than incorrect ones, even though the inflection forms in the sentence are not present in training data. The same applies to text sampling: C2W2C can sample grammatically and semantically correct Finnish sentences character by character.

References

[1] Yoshua Bengio and Jean-Sébastien Senécal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722, 2008.

[2] Piotr Bojanowski, Armand Joulin, and Tomas Mikolov. Alternative structures for character-level rnns. *CoRR*, abs/1511.06303, 2015.

[3] Jan A. Botha and Phil Blunsom. Compositional morphology for word representations and language modelling. *CoRR*, abs/1405.4273, 2014.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[5] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. A character-level decoder without explicit segmentation for neural machine translation. *CoRR*, abs/1603.06147, 2016.

[6] Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. *CoRR, abs/1502.02367*, 2015.

[7] Mathias Creutz and Krista Lagus. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology, 2005.

[8] Seppo Enarvi and Mikko Kurimo. Theanolm-an extensible toolkit for neural network language modeling. *arXiv preprint arXiv:1605.00942*, 2016.

[9] Filip Ginter and Jenna Kanerva. Fast training of word2vec representations using n-gram corpora, 2014.

[10] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. Maxout networks. *International Conference on Machine Learning*, 28:1319–1327, 2013.

[11] Joshua Goodman. Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 561–564. IEEE, 2001.

[12] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, volume 1, page 6, 2010.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997.

[14] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *CoRR*, abs/1602.02410, 2016.

[15] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[17] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005.

[18] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*, 2016.

[19] Krister Lindén, Erik Axelson, Sam Hardwick, Tommi A Pirinen, and Miikka Silfverberg. Hfst—framework for compiling and applying morphologies. In *International Workshop on Systems and Frameworks for Computational Morphology*, pages 67–85. Springer, 2011.

[20] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *CoRR*, abs/1508.02096, 2015.

[21] Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113, 2013.

[22] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and J Cernocky. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 2012.

[23] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

[24] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273, 2013.

[25] David Money and Sarah L Harris. *Digital design and computer architecture*. Morgan Kaufmann Publishers, 2007.

[26] Miikka Silfverberg, Teemu Ruokolainen, Krister Lindén, and Mikko Kurimo. Finnpos: an open-source morphological tagging and lemmatization toolkit for finnish. *Language Resources and Evaluation*, pages 1–16, 2015.

[27] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[28] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197, 2012.

[29] David Vilar, Jan-T Peter, and Hermann Ney. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39. Association for Computational Linguistics, 2007.