Paper 9 | Character-Aware Neural Language Models by Yoon Kim et. al – published on 1st December 2015

Abstract

1. A simple neural language model that relies only on character-level inputs but predictions are still made at the word-level
2. The model employs a CNN and a highway network over characters, whose output is given to a LSTM recurrent neural network language model
3. On English Penn Treebank, the model is on par with existing state-of-art with 60% fewer parameters
4. On languages with rich morphology (Arabic, Czech, French, German, Spanish, Russian), the model outperforms word-level/morpheme-level LSTM baselines (with fewer parameters)
5. Analysis of word representations obtained from the character composition part of the model reveals that the model is able to encode, from characters only, both semantic and orthographic information

Introduction

1. A language model is formalised as a probability distribution over a sequence of strings (words)
2. Traditional methods usually involve making an n-th order Markov assumption and estimating n-gram probabilities via counting and subsequent smoothing. This is simple to train but probability of rare n-grams can be poorly estimated due to data sparsity
3. Neural Language Models (NLM) address the n-gram data sparsity issue through parameterisation of words as vectors and using them as inputs to a NN
    - Weakness of NLMs – they are blind to subword information (e.g. morphemes). For example, they do not know that "eventful", "eventfully", and "uneventfully" should have structurally related embeddings in the vector space
    - This means that embeddings of rare words can be poorly estimated, leading to high perplexities for rare words. This is problematics in **morphologically rich languages** or domains with **dynamic vocabularies (e.g. social media)**

Models

1. Recurrent Neural Network (RNN)
    - At each time step t, an RNN takes the input vector and previous hidden state vector to produce the next hidden state (or current) by applying the following recursive operation

      $$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b})$$

    - In theory, RNN can summarise all historical information up to time t with hidden state h_t. However, learning long-range dependencies with a vanilla RNN is difficult due to vanishing/exploding gradients (due to element-wise multiplication w.r.t. time)
    - **LSTM** addresses the problem of learning long range dependencies by augmenting the RNN with a memory cell vector at each time step.

Technically, one step of an LSTM takes as input x_t, h_{t-1}, c_{t-1} and produces h_t, c_t via the following intermediate calculations:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{U}^i\mathbf{h}_{t-1} + \mathbf{b}^i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f\mathbf{x}_t + \mathbf{U}^f\mathbf{h}_{t-1} + \mathbf{b}^f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^o\mathbf{x}_t + \mathbf{U}^o\mathbf{h}_{t-1} + \mathbf{b}^o) \\
\mathbf{g}_t &= \tanh(\mathbf{W}^g\mathbf{x}_t + \mathbf{U}^g\mathbf{h}_{t-1} + \mathbf{b}^g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

- Memory cells in the LSTM are additive w.r.t. time, alleviating the gradient vanishing problem

2. RNN Language model
3. Character-level CNN
4. Highway Network
    - Simple replace x^k (word embedding) with y^k at each t in the RNN-LM – this simple model performs well on its own

      one layer of a highway network does the following:

      $$\mathbf{z} = \mathbf{t} \odot g(\mathbf{W}_H\mathbf{y} + \mathbf{b}_H) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{y}$$

      where t is the **transform gate**

      1 – t is the **carry gate**
    - Highway layers allow for training of deep networks by adaptively carrying some dimensions of the input directly to the output

Experimental Setup

1. Use perplexity (PPL) to evaluate the performance of the models, which is given by:

   $$PPL = \exp\left(\frac{NLL}{T}\right)$$

    - where NLL is calculated over the test set. We test the model on corpora of varying languages and sizes
2. Optimisation
    - Picking the best model on the validation set
    - Use dropout, with probability 0.5 for regularisation on the LSTM input-to-hidden layers and the hidden-to-output softmax layer
    - Gradient norm constraint is crucial in training the model. The paper contrained the L_2 norm of the gradient to be below 5
    - Employed a hierarchical softmax – a common strategy for training language models with very large vocabulary

Results and Discussions

1. The paper ran two versions of the proposed model to assess the trade-off between performance and size

		Small	Large
CNN	d	15	15
	w	$[1, 2, 3, 4, 5, 6]$	$[1, 2, 3, 4, 5, 6, 7]$
	h	$[25 \cdot w]$	$[\min\{200, 50 \cdot w\}]$
	f	tanh	tanh
Highway	l	1	2
	g	ReLU	ReLU
LSTM	l	2	2
	m	300	650

Table 2: Architecture of the small and large models. d = dimensionality of character embeddings; w = filter widths; h = number of filter matrices, as a function of filter width (so the large model has filters of width $[1, 2, 3, 4, 5, 6, 7]$ of size $[50, 100, 150, 200, 200, 200, 200]$ for a total of 1100 filters); f, g = nonlinearity functions; l = number of layers; m = number of hidden units.

2. Results

	PPL	Size
LSTM-Word-Small	97.6	5 m
LSTM-Char-Small	92.3	5 m
LSTM-Word-Large	85.4	20 m
LSTM-Char-Large	78.9	19 m
KN-5 (Mikolov et al. 2012)	141.2	2 m
RNN† (Mikolov et al. 2012)	124.7	6 m
RNN-LDA† (Mikolov et al. 2012)	113.7	7 m
genCNN† (Wang et al. 2015)	116.4	8 m
FOFE-FNNLM† (Zhang et al. 2015)	108.0	6 m
Deep RNN (Pascanu et al. 2013)	107.5	6 m
Sum-Prod Net† (Cheng et al. 2014)	100.0	5 m
LSTM-1† (Zaremba et al. 2014)	82.7	20 m
LSTM-2† (Zaremba et al. 2014)	78.4	52 m

Table 3: Performance of our model versus other neural language models on the English Penn Treebank test set. PPL refers to perplexity (lower is better) and size refers to the approximate number of parameters in the model. KN-5 is a Kneser-Ney 5-gram language model which serves as a non-neural baseline. †For these models the authors did not explicitly state the number of parameters, and hence sizes shown here are estimates based on our understanding of their papers or private correspondence with the respective authors.

   - Our large model is on par with existing state-of-the-art despite having approximately 60% fewer parameters. Meanwhile, our small model significantly outperforms other NLMs of similar size, even though it is penalized by the dataset already has OOV words replace by <unk>
3. Other languages
   - English is a relatively simple language from a morphological standpoint, hence why the paper focused on other languages
   - On Data-s, the character-level models outperform their word-level counterparts despite being a smaller model
   - The character models also outperform their morphological counterparts
   - The paper used the same architecture for all languages and did not perform any language-specific tuning of hyperparameters
4. Learned word representations
   - By comparing the word representations obtained before and after the highway layers, the paper was able to observe the following:

	In Vocabulary					Out-of-Vocabulary		
	while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
LSTM-Word	*although*	*your*	*conservatives*	*jonathan*	*advertised*	–	–	–
	letting	*her*	*we*	*robert*	*advertising*	–	–	–
	though	*my*	*guys*	*neil*	*turnover*	–	–	–
	minute	*their*	*i*	*nancy*	*turnovor*	–	–	–
LSTM-Char (before highway)	*chile*	*this*	*your*	*hard*	*heading*	*computer-guided*	*informed*	*look*
	whole	*hhs*	*young*	*rich*	*training*	*computerized*	*performed*	*cook*
	meanwhile	*is*	*four*	*richer*	*reading*	*disk-drive*	*transformed*	*looks*
	white	*has*	*youth*	*richter*	*leading*	*computer*	*inform*	*shook*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*	*computer-guided*	*informed*	*look*
	whole	*this*	*your*	*gerard*	*training*	*computer-driven*	*performed*	*looks*
	though	*their*	*doug*	*edward*	*traded*	*computerized*	*outperformed*	*looked*
	nevertheless	*your*	*i*	*carl*	*trader*	*computer*	*transformed*	*looking*

Table 6: Nearest neighbor words (based on cosine similarity) of word representations from the large word-level and character-level (before and after highway layers) models trained on the PTB. Last three words are OOV words, and therefore they do not have representations in the word-level model.

- Before highway layers, the representations seem to solely rely on surface forms (for example, the nearest neighbours of you are your, young, four, youth etc)
- After highway layers, however, the representations seem to enable encoding of semantic features that are not discernable from orthography alone
- The model is also able to correct for incorrect/non-standard spelling

5. Learned character N-gram representations
    - By feeding each character n-gram into CharCNN and use its output as the fixed dimensional representation for the corresponding character n-gram, the model learns to differentiate between prefixes, suffixes and others
6. Highway layers
    - Performance decreases significantly when training a model without any highway layers
    - Having one to two highway layers was important but more highway layers generally resulted in similar performance
    - Having more convolutional layers before max-pooling didn't help
    - Highway layers did not improve models that only used word embeddings as inputs
7. Effect of corpus/vocab sizes
    - The paper also studies the effect of training corpus/vocabulary sizes on the relative performance between the different models
    - The results suggest that the perplexity reductions become less pronounced as the corpus size increases, we nonetheless find that the character-level model outperforms the word-level model in all scenarios
8. Further observations
    - Combining word embeddings with the CharCNN's output to form a combined representation of a word resulted in slightly worse performance. **This is a surprising result**

Related work

1. Neural language models encompass a rich family of neural network architectures for language modelling
    a. Feed-forward
    b. Recurrent
    c. Sum-product

d. Log-bilinear
e. CNN

2. In order to address rare word problem, a paper represented a word as a set of shared factors embeddings. The Factored Neural Language Model (FNLM) can incorporate morphemes, word shape information or any other annotation
3. Another direction of work has been purely character level NLMs, wherein both input and output are characters. Character-level models obviate the need for morphological tagging or manual feature engineering and have the attractive property of being able to generate novel words. **However, they are generally outperformed by word-level models**

Conclusion

1. **Using CharCNN and highway layers for representation learning (as input into word2vec) remains an avenue for future work**

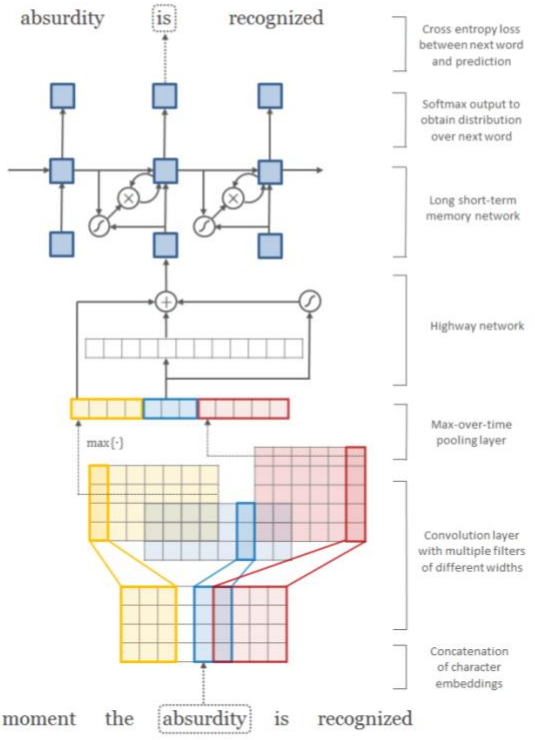

Figure 1: Architecture of our language model applied to an example sentence. Best viewed in color. Here the model takes *absurdity* as the current input and combines it with the history (as represented by the hidden state) to predict the next word, *is*. First layer performs a lookup of character embeddings (of dimension four) and stacks them to form the matrix $\mathbf{C}^k$. Then convolution operations are applied between $\mathbf{C}^k$ and multiple filter matrices. Note that in the above example we have twelve filters—three filters of width two (blue), four filters of width three (yellow), and five filters of width four (red). A max-over-time pooling operation is applied to obtain a fixed-dimensional representation of the word, which is given to the highway network. The highway network's output is used as the input to a multi-layer LSTM. Finally, an affine transformation followed by a softmax is applied over the hidden representation of the LSTM to obtain the distribution over the next word. Cross entropy loss between the (predicted) distribution over next word and the actual next word is minimized. Element-wise addition, multiplication, and sigmoid operators are depicted in circles, and affine transformations (plus nonlinearities where appropriate) are represented by solid arrows.