Character-level Recurrent Text Prediction

Melvin Low
mwlow@stanford.edu

Abstract

Text prediction is an application of language models to mobile devices. Currently, the state of the art models use neural networks. Unfortunately, mobile devices are constrainted in both computing power and space and are thus unable to run most (if not all) neural networks. Recently, however, character-level architectures have appeared that have outperformed previous architectures for machine translation. They are advantageous in that they do not require a word embedding matrix and thus require a lot less space. This project evaluates one recent character-level architecture on the text prediction task. We find that the network performs qualitatively well, as well as achieving perplexity levels close to existing methods on the Brown corpus.

1 Introduction

Text prediction is a forecasting task: guessing the next word in a sequence of text. It is integral to the user experience of mobile users, as good text prediction can increase typing speed and reduce errors. The current state of the art models for the task are language models that use neural networks. Although they perform very well, they often require a lot of memory and computational power–two things that a mobile device does not have. Because of this, mobile devices currently use simple frequency statistics (common ngrams) for prediction. The advantage of this method is that it is fast. The disadvantage is that it cannot take into account the context in which the prediction is being made, unlike some neural network methods.

Recently, there has been exploration in the natural language processing community on using character-level architectures instead of word-level ones. These architectures use significantly less memory, as they do not require a large word embedding matrix. In addition, they are able to utilize subword information such as morphenes. They have achieved state of the art results for machine translation.

A natural question is whether character-level architectures can also be used for text prediction. This paper explores this question. We implement one character-level model that has achieved the aforementioned state of the art results on machine translation, and evaluate it qualitatively and quantitatively on several datasets. Results demonstrate that the character model is able to make useful predictions in many cases, and also achieves perplexity results close to existing methods on a benchmark database.

2 Related Work

Traditionally, text prediction and language models used ngrams [1] [2] [3] [4]. Recently, neural networks have given state of the art performance. Pérez et al. investigated recurrent neural networks [5]. Bengio et al. used multilayer perceptrons [6]. Others have tried purely character-level models, where the input and output are both characters [7]. Character-level only models were found to be generally outperformed by word-level models [8].

Most of these aforementioned approaches used word embeddings to turn words of varying sizes into fixed length vectors. Last year, Kim et al. introduced a character-level model that worked on characters instead of words, eliminating the need for a large word embedding matrix [9]. Despite having fewer parameters, their model gave superior performance than previous word-level models on a variety of tasks, including machine translation [10]. We use their model in this paper for text prediction.

3 Datasets

Table 1: Example input (I) and output (O) sentences.

I: they didn't have the needles he needed , and referred him to a sewing store in the mall .
O: they didn't have the __ he needed , and referred him to a __ store in the mall .
I: the librarian at the reference desk directed him to egypt , and there he spent the afternoon .
O: the __ at the reference desk directed him to egypt , and there he spent the afternoon .
I: mother had a boyfriend from havana , a conga drummer named raul repilado .
O: mother had a __ from __ , a __ __ named __ __ .
I: roy couldn't wait to tell his mother that he'd made an extra fifty bucks that day .
O: roy couldn't wait to tell his mother that he'd made an extra fifty bucks that day .

Three datasets were used for evaluation. For the qualitative evaluation, we used subsets of the Corpus of Contemporary American English (COCA) [11] and the Global Web-Based English (GloWbE) [12] corpus. These two corpuses were combined, and the resulting corpus contained around five million tokens and was used to analyze how our model would perform on contemporary English. For quantitative evaluation, we use the Brown corpus [13], which is often used for benchmarking. This corpus contained around two million tokens.

3.1 Preprocessing

All tokens were made lowercase. Punctuation marks were treated as unique words. For the COCA and GloWbE corpuses, we used an output vocabulary of the 5000 most commonly appearing words. For the Brown corpus, we used 16383 words instead, to make it possible to compare our model against existing methods. Unknown words were replaced with the "__" token.

For input to the character-level model, we simply replaced each character with its ordinal representation from 0-255. There is no word embedding in a character-level model, so we did not replace any words with the "__" token.

We refer to the inputs to the network as *input* sentences, and the target sentences as *output* sentences. Example input and output sentences are provided in Table 1.

4 Methodology

From a probabilistic perspective, text prediction is the task of predicting the next word, y_t, from previous history of words y_{t-1}, y_{t-2}.... In other words, the main problem is to find:

$$\arg\max_{y_t} p(y_t|y_{t-1}, y_{t-2}, ...)$$

This is the type of inference task that recurrent neural networks (RNNs) are particularly well suited for solving. In particular, in this paper, we use a long short-term memory (LSTM), which augments the traditional vanilla RNN formulation with a cell state. Concretely, at each timestep t, the LSTM takes x_t, h_{t-1}, c_{t-1} and produces h_t, c_t via the following calculations:

$$
\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\
o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o) \\
g_t &= \tanh(W^g x_t + U^g h_{t-1} + b^g) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where i_t, f_t, o_t are the input, forget, and output gates. This structure allows gradients to flow additively instead of multiplicatively, hence alleviating problems such as gradient clipping. LSTMs have been used successfully in a wide variety of machine learning tasks. In our model, we used two stacked LSTMs with a hidden state size of 512. We applied batch normalization and dropout to the LSTM outputs, followed by an affine projection to the final output, which was an index within the target vocabulary.

To generate the inputs to the LSTM, we closely followed the approach taken by [9] and [10] to turn variable-sized inputs (words) into fixed-size inputs (embeddings). Specifically, we first converted each character of a word into a character embedding vector of size $D = 32$. We then convolved multiple filters of dimensions (W, D) over the resulting embedding vectors, where W ranged from 1 to 6. Max pooling was used to attain a single number from each filter. These numbers were concatenated to give the final word embedding vector for the input word. Concretely, we used $25 * W$ filters for each W, resulting in a word embedding vector size of dimension:

$$(1 + 2 + 3 + 4 + 5 + 6)(25) = 525$$

[9] found that the embedding method described above already gave a high quality language model. However, they discovered that adding a highway layer after the max pooling, and before the RNN, improved the model further. The highway layer was introduced by [14]. One layer of the highway layer does:

$$z = t \odot g(W_H y + b_h) + (1 - t) \odot y$$

where g is a nonlinearity (we usd RELU) and $y = \sigma(W_T y + b_T)$. We used the same initializations as [9], setting b_T to be small random numbers around -2. RELU was used for the nonlinearity. The output of the highway layer was passed into the LSTM, as mentioned before.

In summary, our architecture was following (from left to right):

Input Word - Character Embedding - Convolutions - Max Pooling - Highway Layer - LSTM - Projection

Please see [9] for a detailed diagram.

Note: [9] found that a multilayer perceptron (MLP) performed worse than a highway layer, so we did not attempt to use a MLP in our model. In addition, [9] found that two highway layers did better better than one on large corpuses, although one layer already gave start of the art results. In this paper we used one layer to reduce space and computation time.

5 Training

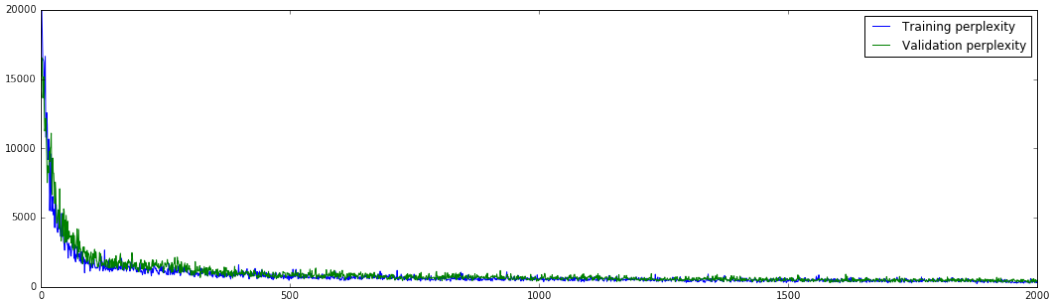

Figure 1: Perplexities over time on the Brown corpus.

We used a 70/20/10 test/train/val split on both COCA+GloWbE and Brown. Training was done with batch sizes of 16, and we truncated backpropation at 32 timesteps. The Adam optimizer was used and initialized with a learning rate of 1e-3. We decayed the rate by a factor of 10, twice, as validation loss stopped decreasing. Finally, dropout was set to 0.5.

On a NVIDIA GTX 980 Ti, training took a full day. Training and validation perplexities on the Brown corpus are shown in Figure 1.

6 Results and Discussion

The char-level model gave decent performance both qualitatively and quantitatively. We provide analyses of both types in the following subsections.

6.1 Qualitative Analysis

Table 2: Sample outputs (correct predictions in red).

“ this discovery is ... important because of the quality of the sample , ” he told reporters .
do not keep your lips stiff and rigid . you have got to keep your lips soft and sensous.
there are three dogs running . two of the dogs are fast . one dog is slow .
why wo n’t you come with me ? my other friend will come with me . he is a very nice man .

The model was able to learn common ngrams, especially bigrams. For example, it learned that the token “the” followed most words, and that “would have been” was a common phrase. There is some indication that the model was more powerful than a purely ngram-based model. For example, it demonstrated limited understanding of tense and plurality through short recall. Some examples are shown below (correct predictions in red):

- i was hungry and i wanted to eat
- we were hungry and we wanted to eat
- i saw her and she was
- i saw her and we were
- i see her and she is
- i see her and we are

The first two examples demonstrate that the model was able to memorize the subject of the sentence, and then repeat it later on. The other four examples demonstrate that the model was able to understand the tense of the verb, “see,” and remember this information (the verb was at the beginning of

the sentence). An ngram model would not have been able to solve this task because of the distance between the verb and the prediction spot. More predictions are given in Table 2.

6.2 Quantitative Analysis

Table 3: Perplexity comparisons on the Brown corpus.

model	train	valid	test
MLP7	210	309	293
MLP9	175	280	276
MLP10 (w/ trigram)	-	265	252
int. trigram	31	352	336
Kneser-Ney back-off	-	332	321
class-based back-off	-	326	312
ours	147	291	320

Table 4: Embedding layer parameters, 525 dim vectors.

input vocab size	params
5000	2625000
10000	5250000
∞ (ours)	19505

We did some quantitative analysis in order to compare our model to existing ones. Specifically, we evaluated perplexities on the Brown corpus. These measurements are given in Table 3, along with measurements of other models. The numbers were taken from [6], which also did he same type of analysis.

Brief examination of the perplexity results shows that our model comes close, but does not beat, other models in terms of test perplexity. Interestingly, it attains better training perplexity than the other models, suggesting that improper regularization may have been a culprit. The best model combines the output of a multilayer perceptron with trigrams, so it is not directly comparable to ours (we could have done the same thing).

However, it is important to note that our character-level model uses far fewer parameters than any word-level model, as shown in Table 4. The character-level model has an effective infinite input vocabulary size, yet uses less than 1/250 of the parameters of a model with input vocabulary of size 10000. This fact, combined with the perplexity results, suggests that our model may be more powerful.

One important thing to note is that, although character-level models use less memory than word-level models, they require more computational time to run because of the convolutions. This is a space-time tradeoff. [9] found that their model was about half as fast as a word-level model. It is possible, of course, to store the 525 dimension vectors generated by the convolutions, but this nullifies the advantages of a character-level model.

7 Conclusion

In this paper we evaluated a character-level language model on the text prediction task. We found that, although the model used significantly fewer parameters than any neural network with a word embedding matrix, it was able to give perplexity results on the Brown corpus close to that of other models. In addition, qualitative analysis showed that the model was able to learn common ngrams, as well as understand (to limited degree) plurality and tense.

The results suggest that character-level models are powerful and may find use in applications where memory is limited, such as in mobile devices. In addition, allowing the model to use more parameters (through more convolutional filters, for example) may improve its performance even more.

References

[1] Claude E Shannon. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64, 1951.

[2] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[3] Steffen Bickel, Peter Haider, and Tobias Scheffer. Predicting sentences using n-gram language models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 193–200. Association for Computational Linguistics, 2005.

[4] Brian Leung and Qi Zhang. Providing relevant text auto-completions, May 21 2007. US Patent App. 11/751,121.

[5] Juan Antonio Pérez-Ortiz, Jorge Calera-Rubio, and Mikel L Forcada. Online text prediction with recurrent neural networks. *Neural processing letters*, 14(2):127–140, 2001.

[6] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[7] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[8] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, and Stefan Kombrink. Subword language modeling with neural networks.

[9] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*, 2015.

[10] Marta R Costa-Jussà and José AR Fonollosa. Character-based neural machine translation. *arXiv preprint arXiv:1603.00810*, 2016.

[11] Mark Davies. The 385+ million word corpus of contemporary american english (1990–2008+): Design, architecture, and linguistic insights. *International journal of corpus linguistics*, 14(2):159–190, 2009.

[12] Mark Davies and Robert Fuchs. Expanding horizons in the study of world englishes with the 1.9 billion word global web-based english corpus (glowbe). *English World-Wide*, 36(1):1–28, 2015.

[13] H Kucera and W Francis. A standard corpus of present-day edited american english, for use with digital computers (revised and amplified from 1967 version), 1979.

[14] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.