INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Experiments in Character-level Neural Network Models for Punctuation

William Gale[1], *Sarangarajan Parthasarathy*[2]

[1]The University of Adelaide, Australia
[2]Microsoft, USA

william.gale@student.adelaide.edu.au, sarangp@microsoft.com

Abstract

We explore character-level neural network models for inferring punctuation from text-only input. Punctuation inference is treated as a sequence tagging problem where the input is a sequence of un-punctuated characters, and the output is a corresponding sequence of punctuation tags. We experiment with six architectures, all of which use a long short-term memory (LSTM) network for sequence modeling. They differ in the way the context and lookahead for a given character is derived: from simple character embedding and delayed output to enable lookahead, to complex convolutional neural networks (CNN) to capture context. We demonstrate that the accuracy of proposed character-level models are competitive with the accuracy of a state-of-the-art word-level Conditional Random Field (CRF) baseline with carefully crafted features.
Index Terms: speech recognition, punctuation prediction, neural networks

1. Introduction

Automatic speech recognizers typically output a sequence of words with no punctuation marks. Restoration of punctuation is important for enhancing human readability as well as improving the accuracy of post-processing such as language understanding or translation. While acoustic features are necessary for accurate punctuation prediction, results obtained with only textual features are often useful[1]. Conditional Random Fields (CRF)[1], and more recently Recurrent Neural Networks (RNN)[2] and Convolutional Neural Networks (CNN)[3], produce state-of-the-art results. All of these systems use word-level features: n-grams or embeddings.

In this paper, we explore character-level neural network models for punctuation prediction. The benefits of modeling at the character level are:

- simple tokenization: input and output are simply a sequence of utf8 tokens which makes it easy to accommodate languages with no natural word boundaries,
- no necessity to handle out-of-vocabulary tokens,
- easy to extend to further punctuation characters, such as hyphens, quotes, and accents.

A disadvantage is that much larger contexts need to be modeled at the character-level: a factor of about 6, the average number of characters in a word.

Input to the punctuation model may be a word sequence corresponding to a single sentence, or a paragraph consisting of multiple sentences. The fraction of utterances which are single sentences is a function of the application: short-message dictation tends to be mostly single sentences whereas multiple sentence input is common when creating large documents or transcribing talks. Inferring punctuation for single sentences is not as simple as it may seem for the following reasons:

- Speech recognition errors and speech disfluencies lead to an increase in punctuation errors[1]
- Question-mark prediction depends on long-distance triggers which makes it difficult to model for hidden-unit language models[4] and CRFs: *hey,* ***do you*** *want to play tennis if you get here before dark?*
- Informal language which may not be well-represented in the training data: *was running late.* vs. *i was running late.*
- Almost any statement can be turned into a question by changing the intonation: *[i] want to play tennis.* vs. *[do you] want to play tennis?*, where the words in brackets are implied and conveyed by intonation. While acoustic features are necessary to handle such cases, the above example is more likely a question than a statement, and a text-only model could learn to make the most probable prediction.

2. Punctuation modeling

2.1. Problem description

We first focus on the simpler task of inferring punctuation on single sentences–specifically commas, periods and question marks. We then generalize these models to the task of inferring punctuation on multiple sentences, a much harder task as it also requires the models to learn sentence segmentation.

Input to the models constitutes the raw, unformatted, text alone, with each model outputting a series of tags $\{O,<.>,<?>,<,>\}$, where there is a one-to-one mapping between the input characters and the output tags. A further design decision was made to align the punctuation tags with spaces (and hence word-boundaries) due to the convenience that it provides when adding additional punctuation features, such as accents. Aligning the punctuation with spaces did not provide any performance gains over the unaligned task.

2.2. Baseline

Our baseline system uses a linear-chain CRF[5, 1] with the following word-level features:

- Word-shape: alpha-digit, character-ngram word suffix
- Word n-grams, word class (Brown cluster label)
- Language model features
- Word N-gram triggers extracted from training data: select n-grams with the largest point-wise mutual information $\frac{P(ngram\_in\_sent,punc=p)}{p(ngram\_in\_sent)p(punc=p)}$ where $sent$ represents a window of words around the punctuation mark. This feature is not commonly used in CRF models for punctuation but provides significant improvement in accuracy of question-mark prediction.

http://dx.doi.org/10.21437/Interspeech.2017-1710

2.3. Neural Network Architectures

2.3.1. Delay Model

All of the neural network language models explored in this paper use LSTM models due to their ability to model long term dependencies[6, 7] and adapt to variable length sequences[6]. The models were trained using the cross-entropy criterion.
The Delay model follows an approach that has been used at a word-level for Part of Speech tagging in the literature[8], however we make a notable and significant change in that we introduce a time delay of size n between an input x_t and its corresponding output y_t. This time delay provides the model with more context before it makes its predictions–roughly allowing the model to capture n-grams, centered on the current character, of size $2n + 1$. The delay was introduced by post-padding the source sequence with one <*EOT*> token and $n - 1$ <*PAD*> tokens and pre-padding the target sequence with n <*PAD*> tokens. See Figure 1a.

2.3.2. N-gram Model

The N-gram model allows character n-grams to be provided to the model directly, with the intention of reducing the complexity of the task that the LSTM has to learn. Character n-grams were fed into the LSTM layers as a concatenation of the n character embeddings centered on the t^{th} character $(e_{t-\lfloor \frac{n-1}{2} \rfloor}, ..., e_t, ..., e_{t+\lceil \frac{n-1}{2} \rceil})$ where there was no delay between any input x_t and its output y_t. This method was modeled on the procedure that was used in the CRF baseline, but using characters instead of words. The source sequence is post-padded with a single <*EOT*> token to denote the end of a passage of text. See Figure 1b.

2.3.3. CNN Feature Model

The CNN-Feature model is similar to an approach recently introduced by others[9] and attempts to capture rich multi-character embeddings through the use of a single CNN layer with multiple kernel sizes. While the original model uses pooling after each convolutional layer to capture n-grams of these embeddings, our CNN-Feature model is conceptually simpler and omits the pooling layer–instead feeding the multi-character embeddings straight to the LSTM layers. The source sequence is post-padded with a single <*EOT*> token, but is also now both pre-padded and post-padded with <*NULL*> tokens so each kernel width produces T outputs–retaining a one-to-one mapping with the output sequence. As was found in the original work, adding a *highway layer*[10] after convolutions was found to improve performance significantly. See Figure 1c.

2.3.4. CNN Feature N-gram Model

The CNN-Ngram Model more closely follows the work of others[9] and includes a temporal max-pooling layer of width n after each convolutional layer to capture n-grams of the convolutional character features. Our approach has one key difference to the original work in that we pad n-gram regions that fall inside the text with the characters to either side, instead of zeros. We believe that our approach is more natural and it was found to improve performance on our task. See Figure 1d.

2.3.5. Dilated CNN Model

The Dilated CNN model used an architecture that has recently achieved excellent performance on images[11], audio[12] and text[13]. This architecture captures large contexts cheaply by stacking multiple layers of *atrous* convolutions (convolutions with holes). In our implementation each layer captures twice the context of the previous layer, while still providing unique character-level features. We used five dilated CNN layers to capture character n-grams of width 32 which were then input to the LSTM layers. *Residual connections*[14] between convolutional layers were found to be essential for this approach to work, with gated residual outperforming simple summation. Using a highway layer to transform each character's captured features after every convolutional layer was found to further improve performance. We only used the final layer's output as input to the LSTM, as concatenating the output from all layers provided negligible gains in performance. See Figure 1e.

2.3.6. Bidirectional-LSTM Model

The Bidirectional-LSTM Model[15, 8] uses LSTM to traverse the source sequence in both directions to generate two hidden states, $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, for each input character x_t. This allows the model to use the entire text as context when predicting the punctuation, resulting in a more powerful model. However, the bidirectional model is more expensive and requires the full text to be available for inference–for these reasons this paper focuses on the previous five models.

3. Experiments

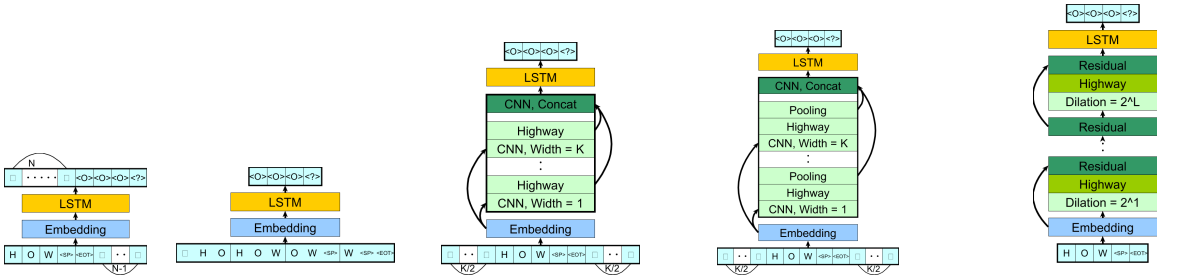
3.1. Datasets

3.1.1. Newswire text and Synthetic Data

The first set of experiments establish whether character-level neural network models are competitive with word-level models. These experiments then explore a simple method for reducing the impact of a domain mismatch between training and evaluation data. The word-level and character-level models were trained on newsfeed data that was augmented with a synthetic dataset to better capture the informal style of spoken utterances. Questions represented 18.9% of the combined set of around 1,200,000 training instances. Models were evaluated on a separate dataset constructed from short message dictation (SMD) output that had been manually annotated by language experts–for the domain mismatch experiments half of this data was added to the training and validation sets, while the other half formed a new evaluation set. Results are broken down into the individual precisions and recalls for periods, question marks and commas, as well as the overall F-measure, because the commas are much harder to model due to stylistic variations and inter-annotator disagreement.

3.1.2. Matched Webcrawl Data

The next set of experiments sought to provide a fair comparison of the character-level architectures, removing the impact of any domain mismatch from consideration. All of the character models were evaluated in a matched setting using data crawled from the web. Training used 2,000,000 examples, while the cross validation and evaluation were performed on two hold-out sets containing 20,000 examples each–all sets were 50% questions. Once again, we report the individual precisions and recalls for periods, question marks and commas, as well as the overall F-measure.



(a) *Delay* (b) *N-gram (3-gram)* (c) *CNN-Feature* (d) *CNN-Ngram* (e) *Dilated CNN*

Figure 1: *The Neural Network Model Architectures*

3.2. Hyper-Parameters and Preprocessing

The hyper-parameter settings for all of the models were individually optimized to maximize their performance on the hold-out cross validation sets without any consideration of overall model sizes and capacities. The difference in performance within the set of best performing hyper-parameter settings on both datasets was within the realm of noise, hence the intersection of these sets was used for all experiments for simplicity.

All models used an input embedding size of 15 and dropout. The delay model used a delay of 8, three layers of 200-dim LSTM and a keep probability of 0.8 between the Embedding-LSTM, LSTM-LSTM and LSTM-Softmax layers. The N-gram model used n-grams of size 20 with the other parameters the same as the delay model. The CNN-Feature model used a total of 1650 kernels (ranging in width from 1 to 7), two layers of 100-dim LSTM and a keep probability of 0.9 between the Embedding-CNN and 0.6 between the CNN-LSTM, LSTM-LSTM and LSTM-Softmax layers. The CNN-Ngram model used temporal max-pooling with a width of 4, but was otherwise the same as the CNN-Feature model. The Dilated CNN model used five dilated CNN layers with 32 kernels per layer, two layers of 128-dim LSTM and a keep probability of 0.9 between the Embedding-CNN, 0.8 between the CNN-CNN and 0.6 between the CNN-LSTM, LSTM-LSTM and LSTM-Softmax layers. The Bidirectional-LSTM model used three layers of 200-dim LSTM and a keep probability of 0.8 between the Embedding-LSTM, LSTM-LSTM and LSTM-Softmax layers.

All data was converted to lowercase, individual digits were normalized to zero (while retaining number length) and non-utf8 characters were normalized by either substituting them with a utf8 equivalent or removing them. All commas, question marks and end-of-sentence periods were removed from the source sequence and appeared only in the target sequence. The text was tokenized for the word-level models, while space characters were treated as another normal input character for the character-level models.

3.3. Results

3.3.1. Baseline

The results for word-level CRF, word-level RNN with a delay of 1 and word-level RNN with n-grams of 3 (all capturing roughly equivalent contexts at each step) are compared with the character-level RNN delay model in Table 1 and Figure 2a. The character-level model can be seen to perform competitively with all the word-level models.

Table 1: *Character-level vs Word-level models on SMD*

Model	Period Precision	Period Recall	Question Precision	Question Recall	Comma Precision	Comma Recall	Overall F1
Word CRF	0.953	0.977	0.907	0.831	**0.539**	0.264	**0.873**
Word Delay-RNN	0.958	0.965	0.867	0.852	0.463	**0.423**	0.864
Word Ngram-RNN	**0.959**	0.947	0.813	**0.858**	0.448	0.411	0.851
Char Delay-RNN	0.949	**0.987**	**0.943**	0.803	0.471	0.383	0.872

3.3.2. Adding in-domain data

The impact of adding in-domain data during the training of the character-level delay model is presented in Table 2. The addition of this small amount of in-domain data ($\approx 0.3\%$ of the total training data) can be seen to provide a notable improvement in the performance across all metrics.

Table 2: *Adding in-domain (SMD) data to training*

Training Data	Period Precision	Period Recall	Question Precision	Question Recall	Comma Precision	Comma Recall	Overall F1
Newswire + DSAT	0.949	0.987	0.943	0.803	0.471	0.383	0.872
Newswire + DSAT + SMD	**0.953**	**0.991**	**0.956**	**0.816**	**0.505**	**0.463**	**0.881**

3.3.3. Matched Single Sentence

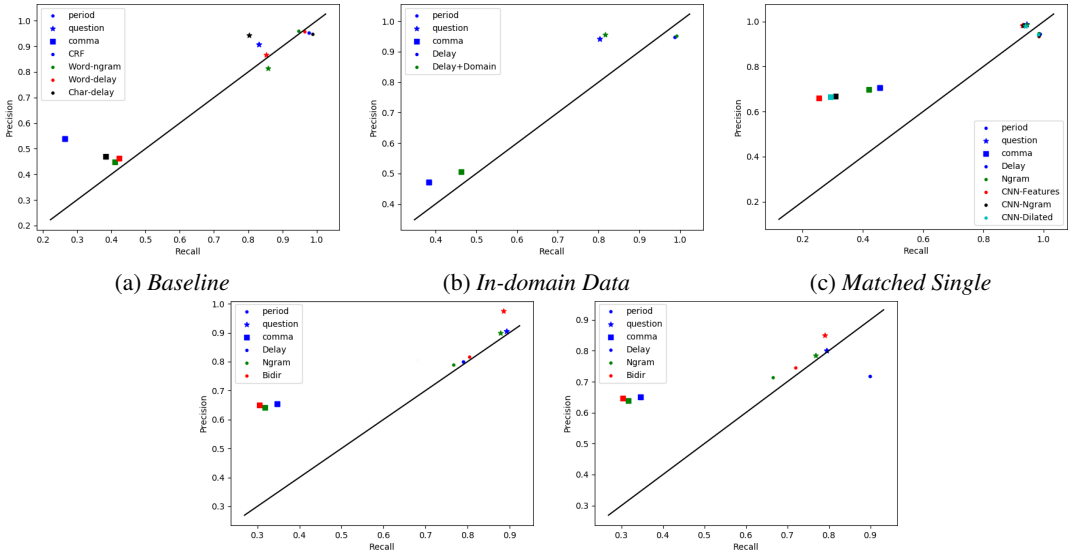
The performance of the Delay, N-gram, CNN-Feature, CNN-Ngram, Dilated CNN and Bidirectional-LSTM models on the matched single-sentence task are presented in Table 3 and Figure 2c. These experiments show that all models achieve similar performance, with the Bidirectional-LSTM model performing the best, closely followed by the simplest Delay model.

Table 3: *Matched single sentences*

Model	Period Precision	Period Recall	Question Precision	Question Recall	Comma Precision	Comma Recall	Overall F1
Delay	0.946	**0.989**	**0.988**	0.944	**0.707**	**0.457**	0.871
N-gram	0.947	0.985	0.985	0.944	0.697	0.42	0.865
CNN-Feature	0.934	0.984	0.983	0.930	0.661	0.253	0.837
CNN-Ngram	0.937	0.984	0.983	0.933	0.668	0.309	0.845
Dilated CNN	0.936	0.983	0.983	0.933	0.664	0.295	0.843
Bidir-LSTM	**0.962**	0.986	0.986	**0.959**	0.658	0.449	**0.880**

3.3.4. Matched Multiple Sentence

The performance of the Delay, N-gram and Bidirectional-LSTM models on the matched multi-sentence task are presented on the whole-text task in Table 4 and Figure 2d and the mid-text task in Table 5 and Figure 2e. These results show that the mid-sentence task is much harder–namely because the model cannot use the end-of-text token when making these predictions. The comma performance remained constant between the whole and mid-text tasks, demonstrating that the models never place commas at the end of the text and that they often choose to combine



(a) *Baseline* (b) *In-domain Data* (c) *Matched Single*

(d) *Matched Multiple (whole)* (e) *Matched Multiple (mid)*

Figure 2: *The Neural Network Model Results*

Table 4: *Matched multiple sentences–whole text*

Model	Period Precision	Period Recall	Question Precision	Question Recall	Comma Precision	Comma Recall	Overall F1
Delay	0.800	0.790	0.906	**0.893**	**0.650**	**0.346**	0.737
N-gram	0.789	0.767	0.899	0.879	0.639	0.316	0.722
Bidir-LSTM	**0.817**	**0.804**	**0.976**	0.885	0.646	0.303	**0.740**

Table 5: *Matched multiple sentences–mid text*

Model	Period Precision	Period Recall	Question Precision	Question Recall	Comma Precision	Comma Recall	Overall F1
Delay	0.717	**0.899**	0.800	**0.795**	**0.650**	**0.346**	0.633
N-gram	0.713	0.666	0.784	0.768	0.639	0.316	0.611
Bidir-LSTM	**0.745**	0.720	**0.850**	0.790	0.646	0.303	**0.638**

separate questions and sentences into a single phrase. The models prefer to preserve context within a phrase than split it.

4. Discussion

The character-level models achieve impressive results that are on-par with state-of-the-art CRF models–without the need for extensive hand-engineered features. The character-level models also successfully infer punctuation on sequences of up to 150 characters in length, indicating that plain Vanilla-LSTM is capable of capturing very long-term dependencies–to a much greater extent than we thought was possible. The similarity in the performance of all the models on the matched dataset indicates that inferring punctuation from text for single sentences is approaching the error floor imposed by ambiguities in the text due to the lack of acoustics and variability in commas. Inferring punctuation on the multi-sentence scenario is more challenging, but preliminary results are encouraging. We expect that the frequent joining of related questions and sentences into longer phrases by the multiple sentence model is likely due to the way that the training data was constructed (randomly combining both related and unrelated phrases) and is something that we will investigate further. It is also worth noting that applying the more powerful CNN+RNN language models to this domain was non-trivial as all models we explored were outperformed by the simple Delay model, despite providing performance gains on many other language modeling tasks[9, 16].

Adding a small amount of in-domain data to the training set ($\ll 1\%$) was found to provide an easy way to overcome some of the impact of the domain-mismatch between the training and evaluation sets. It is unclear whether adding this additional data improved performance by shifting the models' convergence to better local optima, or whether the models "learnt" from the data–however it provided a simple way of improving performance across all measured metrics on our task, potentially making it a useful tool to quickly adapt models to new domains.

5. Conclusion and Future Work

We have demonstrated that character-level neural networks achieve results on-par with state-of-the-art CRFs when inferring punctuation on raw text alone. Issues arising from factors such as ambiguity, poor grammar, disfluencies and SMD reco errors make this a difficult task and our experiments show that we're approaching the error floor when processing recognition output. Future work will need to incorporate additional context, such as acoustic features, to achieve significant gains–despite the challenges posed in training such models. We also found that adding small amounts of in-domain data provided gains across all measured metrics and helped reduce the impact of a domain mismatch between the training and evaluation sets. We have shown that character-level models achieve remarkable results on the single-sentence task and that we are also making progress on the more challenging multiple-sentence task. Our results demonstrate that character-level neural network models are incredibly powerful and their use on both this, and related, tasks is worthy of further exploration.

6. Acknowledgments

This work was completed during an Internship at Microsoft and would not have been possible without their support.

7. References

[1] N. Ueffing, M. Bisani, and P. Vozila, "Improved models for automatic punctuation prediction for spoken and written text," in *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pp. 3097–3101, 2013.

[2] O. Tilk and T. Alumäe, "Bidirectional recurrent neural network with attention mechanism for punctuation restoration," in *INTERSPEECH 2016*, pp. 3047–3051, 2016.

[3] X. Che, C. Wang, H. Yang, and C. Meinel, "Punctuation prediction for unsegmented transcript based on word vector," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)* (N. C. C. Chair), K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, eds.), (Paris, France), European Language Resources Association (ELRA), may 2016.

[4] A. Gravano, M. Jansche, and M. Bacchiani, "Restoring punctuation and capitalization in transcribed speech," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4741–4744, 2009. doi:10.1109/ICASSP.2009.4960690.

[5] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), pp. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[7] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *arXiv preprint arXiv:1503.04069*, 2015.

[8] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[9] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *arXiv preprint arXiv:1508.06615*, 2015.

[10] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," *arXiv preprint arXiv:1507.06228*, 2015.

[11] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *In International Conference on Machine Learning (ICML)*, 2016a.

[12] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu., "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016b.

[13] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[15] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," in *International Joint Conference on Neural Networks*, 2005.

[16] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.