Creating a Simplifier

Amol Kapoor
Department of Computer Science
Oxford University

Bernardo Orozco
Department of Computer Science
Oxford University

Abstract

Text simplification aims to make complex information easier to understand and therefore available to more people. Traditional methods of text simplification rely on grammar parsing rules based on intuitions about how text is grouped, or substitutions of complex words based on pre-specified synonym dictionaries. Both methods are unsuitable for generalized tasks and scale poorly. This project aims to create a generalized text simplifier using a deep learning seq2seq model. The model will be trained on a combination of Wikipedia-SimpleWikipedia sentence pairs. Our model achieves a BLEU score of 0.53499. Qualitatively, our model is able to generate approximate translations of unseen Wikipedia documents. We suggest examining larger models, character-based architectures, and training with multi-length inputs for future work.

1 Introduction

1.1 Early Research

The goal of text simplification is to take a source sentence and create a target sentence that is less complex in word choice, less grammatically ambiguous, and generally easier to read. Text simplification is an important mechanism to make information easier to access, and therefore has many important applications across numerous fields.

Text simplification is a rich field with a long history; however, there is relatively little penetration of deep learning methods in the field, with most research involving statistical or rule based methods [1]. Though these methods can be give good results, they are highly domain specific as they depend on significant amounts of data and hand crafted rules dependent on prior knowledge. Neural language models can make up for the shortcomings of statistical or rules based models.

1.2 Neural Language Models

Bengio et. al. (2003) [2] implemented one of the earliest neural language models as a means to handle these limitations. It was proposed that neural models could perform better than statistical models by learning distributed feature vectors for each word in the vocabulary organically, and then learning a probability function over those words for various sequences. This early feedforward network used fixed length vectors for the vocabulary input. It performed better than the state-of-the-art levels, but only scaled linearly with the vocabulary. This was a marked improvement over the exponential growth of statistical models.

Feedforward networks struggle to learn sequences because there is no internal representation of past data. The requirement for fixed length representations limits many key applications of language modeling where the input or output length is not known a priori. In order to tackle these problems, Mikolov et. al. (2010) [3] proposed using recurrent neural networks (RNNs) instead of feedforward networks. Though this architecture proved more difficult to train for long term dependencies than n-gram models, it resulted in significant reduction in word error rate for major language model tasks.

1.3 Machine Translation

We examine Machine Translation as a similar field to Text Simplification, as both tasks focus on sequential data encoding conversion with minimum loss of information. Improvements have been made with neural networks in the field of Machine Translation that could be applied to text simplification. A few relevant developments are summarized below.

The development of encoder-decoder architectures [4] [5] [6] led to significant improvements for neural networks in Machine Translation tasks, especially when compared to previous statistical or rules based baseline models. In encoder-decoder models, the encoder is a network that takes an input string in the source language and maps it to a feature embedding vector, and the decoder is a network that takes a feature embedding vector and maps it to an output string in the target language. This setup seeks to teach the network how to identify the core meaning of a given sentence by embedding it in a feature space, allowing the decoder to translate the input appropriately. Unlike previous models, these neural models can train all parameters jointly and end-to-end. Experiments with the encoder-decoder architecture found that deep LSTM layers with reversed inputs during training [6] led to better results in sequential data tasks.

A significant problem with Machine Translation models is the requirement for fixed-length input and output vectors. Bahdanau et. al. tackles this problem by introducing an attention mechanism [7]. Instead of relying on fixed length vector inputs, Bahdanau trains the encoder-decoder model to align the source and target sentence and translate simultaneously. Outputs are conditioned on 'annotations' that represent an input word and its surroundings, which are weighted based on alignment scoring. The alignment model can backpropagate cost, allowing joint training. This attention mechanism is later developed further by Vinyals et. al. [8]. The attention mechanism proposed by Vinyals is described below. A longstanding problem with Machine Translation efforts is the inability for language models to deal with rare words or words that are not present in the training data vocabulary. Jean et. al. (2014) [9] propose a sample softmax calculation that is based on importance sampling. By taking only a small number of samples, the proposed algorithm was able to compute the normalization constant (used in softmax calculations) with a small subset of a much larger vocabulary. At each update, only vectors associated with the sampled words are updated. The smaller sample and faster softmax calculation removes limitations on target vocabulary size (testing was done on vocabularies of up to 500k words), while empirically matching full softmax equivalents.

1.4 Applications to Text Simplification

It has been proposed that text simplification can be viewed as a monolingual Machine Translation task [10]. Preliminary research shows that encoder-decoder models traditionally used for Machine Translation are successful at learning simplification rules, such as replace, reorder, and sort [11]. It is worth mentioning that Text Summary also has a significant body of work with deep learning applications, and also shares many important goals with text simplification. Recent studies in summary build on the same encoder-decoder model discussed above with significant success [12][13] compared to past statistical work, likely because text simplification also benefits from training a model that attempts to capture sequential meaning. Across our survey of current research, the encoder-decoder model is state-of-the-art. Thus, we propose using a similar model for text simplification.

2 Model

2.1 Encoder-Decoder Model

In the encoder-decoder model, the encoder converts a (reversed) input sentence composed of a set of word-vectors $\mathbf{x} = (x_1, ..., x_T)$ into a single feature vector c. We define the encoder hidden state h_t at time t as being dependent on input x_t and the previous hidden state h_{t-1}:

$$h_t = f(x_t, h_{t-1})$$

We also define the vector c as dependent on all previous hidden states, as part of the RNN structure:

$$c = q(h_1, ..., h_t)$$

Like Sutskever [6], we use mutlilayered LSTMs to define both f and q. The decoder is trained to predict the next word given feature vector c and all previously predicted words $y_1, ..., y_{t-1}$. With RNNs, the conditional probability is modeled as:

$$p(y_t|y_1, ..., y_{t-1}, c) = g(y_{t-1}, s_t, c)$$

where s_t is the decoder hidden state.

2.2 Attention

Our attention model is based on the work of Vinyals et. al. (2015) [8]. Attention is trained as part of the learning/update sequence of the model; thus, the model learns how to align data without previous annotations. The alignment scheme can be viewed as a feedforward network that determines where to weight attention, with weights being applied to previous hidden states (which roughly corresponds to where in the sentence the encoder has read up to). At time t, attention is calculated with the following equations over input words (1,...,T_A):

$$u_i = v^T tanh(W_1 h_i + W_2 d_t)$$

$$a_i = softmax(u_i)$$

$$d_t^* = \sum_{i=1}^{T_A} a_i h_i$$

In the above, vector v and matrices W_1, W_2 are learnable parameters, and a_i represents the attention mask applied to the input sentence at time t. The decoder hidden state d_t is concatenated with d_t^* to predict the next word. The final model can be seen in Figure 1. More on attention can be found in Vinyals et. al. (2015) [8].

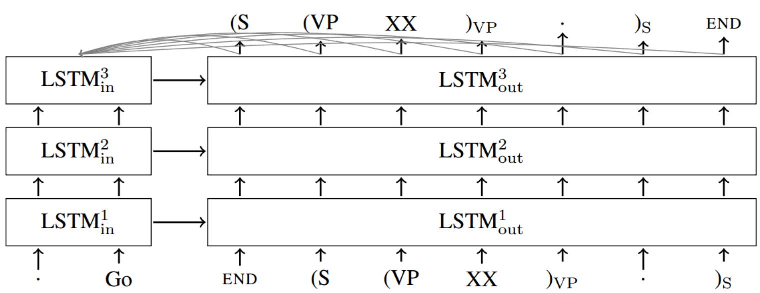


Figure 1: Visualization of the sequence to sequence alignment model used in Vinyals et. al. [8].

2.3 Other Considerations

To avoid overfitting we implement standard regularization procedures, including LSTM unit dropout and L2 regularization on LSTM internal weight parameters. Early experiments resulted in exploding gradients; thus, we added gradient clipping to the network. Increased vocabulary sizes resulted in massive slowdowns on our hardware. As a result, we implement the sample softmax calculation proposed by Jean et. al. [9].

3 Experiment

3.1 Data

Deep learning requires large data sets for neural network models to learn during supervised training. Previous research has used sentence-aligned data sets from Wikipedia and its sister site, Simple Wikipedia [14] [15]. Previous research has also used SimpleWikipedia revisions that are marked

Table 1: Unchanged Network Parameters

Name	Value
Input Sentence Length	35
Output Sentence Length	50
Vocabulary Size	80000
Batch Size	64
Softmax Sample Size	512
Base Learning Rate	1.0
Decay Factor	0.99
Dropout Probability	0.5
Max Gradient Norm	5.0
Layers	3

as simplifications [14], and document-aligned data sets from Wikipedia and Simple Wikipedia [15]. Combined, these data sets present almost 300k pairs of sentences and more than 60k pairs of documents. By our searching, these are the largest publicly available data sets for text simplification. These data sets were compiled between 2010 and 2011; additional data can also be scraped from Wikipedia/SimpleWikipedia for more recent articles, though we did not do so. For this experiment, we do not use the document aligned data to train.

3.2 Setup

The model was built using Tensorflow 1.0 without CUDA, specifically using the seq2seq attention model located in *tf.contrib.legacy_seq2seq.embedding_attention_seq2seq*. Training and testing was done on an ASUS ROG GL551J. Of the 300k Wikipedia/SimpleWikipedia sentence alignments available, we used 10k for the validation set and 10k for the test set. Unchanging network parameters are displayed in Table 1. For hyperparameters we examined the number of units per layer and the L2 regularization constant. We quantitatively test how well our model is able to generalize on sentences, and qualitatively test how well our model is able to simplify complex documents like full Wikipedia pages sentence by sentence. Training was done with simple back propagation using a gradient descent optimizer. We examined nine different architectures with varying regularization constants and layer units using a single cross validation split. For each architecture, we examine perplexity and BLEU score. Architectures were trained for at least 5k steps. Training was done with simple back propagation using a gradient descent optimizer.

3.3 Results

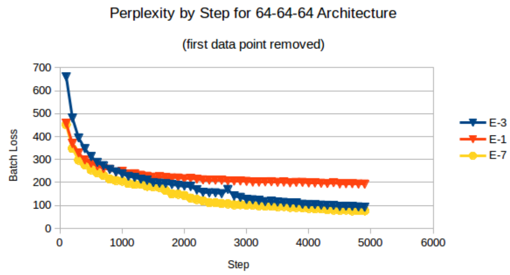
3.3.1 Quantitative

The results of our experiments for hyperparameters can be seen in Figure 2 and Figure 3. Both show that the architecture with the least regularization and most units performs best. In our case, this architecture has 256 units per layer with a regularization constant of 1E-7. The average BLEU score for this architecture was 0.5349982667 (the BLEU scores of our various tested architectures are shown in Table 2). It is worth noting that none of the models we tested had reached convergence by the time 5k steps was reached, as loss was still steadily decreasing with little or no learning rate decay. This result suggests that the model we used likely does not cover all of the expressiveness of the data, and a larger model (i.e. more units) may perform even better.

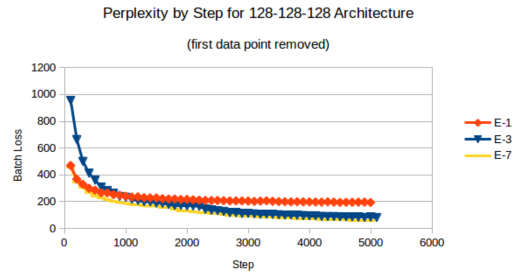
While there are no controls to test our model against (a statistical text simplification model as a control would be the ideal), an average BLEU score greater than 0.5 shows significant overlap with the actual simplified sentence.

3.3.2 Qualitative

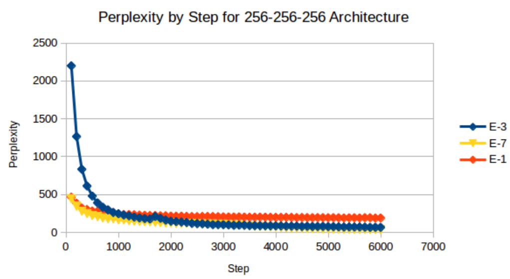
For our qualitative tests, we use the 256-256-256 architecture with a regularization constant of 1E-7 on a random Wikipedia article - in this case, the first article in the document aligned data set for the Polish town of Szczecin. This article contained a high number of previously unseen tokens. An excerpt of the original document and the translation is given below:



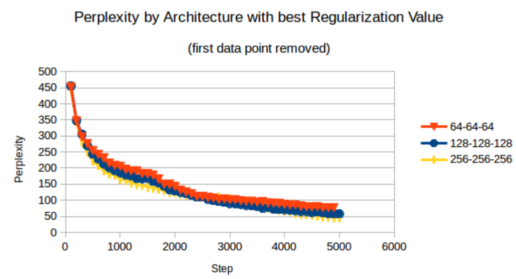
(a) 2a



(b) 2b

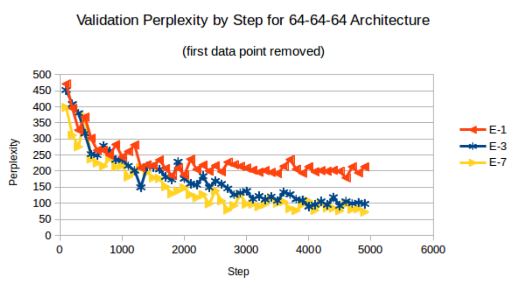


(c) 2c

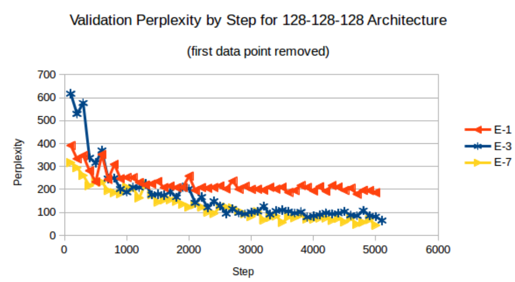


(d) 2d

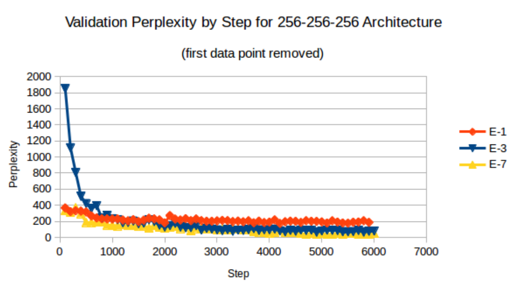
Figure 2: Hyperparameter plots. Plots a-c examine how various levels of regularization perform over time with measurements of perplexity on training sets. Plot d compares the best of each of the architectures from a-c.



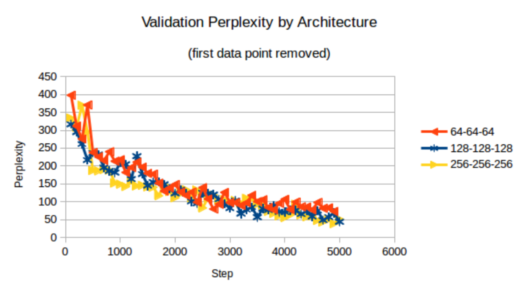
(a) 2a



(b) 2b



(c) 2c



(d) 2d

Figure 3: Hyperparameter plots. Plots a-c examine how various levels of regularization perform over time with measurements of perplexity on validation sets. Plot d compares the best of each of the architectures from a-c.

Table 2: BLEU Scores: Regularization Constant by Units per Layer

	256	128	64
1E-1	0.3730467241	0.3862553793	0.4216232414
1E-3	0.4561883667	0.5294605	0.5004509333
1E-7	**0.5349982667**	0.5119986333	0.4971316333

For other meanings , see Szczecin -LRB- disambiguation -RRB- and Stettin -LRB- disambiguation -RRB- .
Szczecin , is the capital city of the West Pomeranian Voivodeship in Poland .
It is the country 's seventh-largest city and the largest seaport in Poland on the Baltic Sea .
As of June 2009 the population was 406,427 .
Szczecin is located on the Oder River , south of the Szczecin Lagoon and the Bay of Pomerania .
The city is situated along the southwestern shore of DÄbie Lake , on both sides of the Oder and on several large islands between the western and eastern branches of the river .

Other species are two , two -LRB- UNK_ID -LRB- UNK_ID -RRB- -RRB- -LRB- -RRB- .

The city is the county of the county of the United States .
It is located in the UNK_ID and and the largest River .
As of 2000 , it was UNK_ID .
UNK_ID is located located in the east of the United States .

The town is a city of the city of the UNK_ID .

The quality of this translation is fairly poor. However, there are clear indications that the model is indeed learning some generalizations. The sentence structures in the translation are far simpler than their original counterparts. Further, general themes about each sentence make it through the model - for example, the third sentences relate to bodies of water, and the fifth and sixth sentences broadly relate to position/location. Finally, it is worth noting that the model is learning some form of grammar, evidenced by the fact that the translated sentences are in fact readable. These consistencies would suggest that the principle of applying deep sequential neural models is sound. More data and more training time, as well as more complex models, may be useful for improving translations.

4 Conclusion and Future Work

In this paper we developed a sequence to sequence model using state of the art Machine Translation architectures and applied it to the task of Text Simplification. Though our results left much to be desired, they indicated soundness of principle. We propose that increased model expressivity can be achieved with larger models (more layers, more units per layer) and can lead to better results.

The work done in this paper was more proof of concept than experiment. Given more time and resources, a full experiment would include a statistical machine simplification model to act as a control. The statistical model would be trained on the same dataset, and its BLEU scores would be compared to the BLEU scores of our model. A more rigorous experiment would also include better cross validation (with multiple different training/validation/test splits) and more hyperparameters (such as different optimizers, or using different numbers of units within each layer).

From an architecture perspective, there is Machine Translation research that suggests our model can be improved. As of right now, we are using a simple single directional encoder. However, there is evidence to support a bidirectional encoder. Bidirectional encoders are capable of looking forward and backward in time, allowing better representations of context and giving stronger predictive power [7].

We are also currently using a word-based vocabulary system. Character models that break up individual words and attempt to predict the next character have shown improvements over word based systems, especially when handling rare words [16] [17] [18]. Solving the rare word problem would be particularly useful for our model, given our qualitative document simplification results. Given more time, a better architecture might include a convolutional network over characters that is fed into the sequence model we described above.

Finally, we note that attempting to simplify documents using a sentence-to-sentence simplifier is not an intuitive representation of how humans summarize documents. Humans will use the context of an entire document to inform summaries of individual sentences, including whether or not to keep a sentence in the simplified version, or to split one sentence up into multiple sentences; however, simplification is still done on an individual sentence-by-sentence basis. Thus, a good model needs to be able to translate both documents and sentences. Our model cannot handle these tasks in its current state. Current architectures will pad input and output data to the longest length[1]. For documents, the length of the longest representative word vector encoding could be in the thousands, whereas the average length of an equivalent vector for sentences is in the tens. Attempting to train on the same padded data would result in early parts of the network learning mostly sentence alignments, while only later parts of the network learn document alignments. This means that it is impractical to train on document aligned data and sentence aligned data at the same time, even when the data is available, as the network will not receive the same amounts of training in different places.

While some recent work has been done on hierarchical attention models that attend to different lengths of text at the same time [19], we propose a simpler mechanism for training on both document and sentence aligned data simultaneously. During training, randomly pad the data such that the end length is the same, but the text appears somewhere between an amount of forward padding and backward padding[2]. This would allow all parts of the network to see document length and sentence alignments during training. We theorize that such a training system would allow the network to learn how to simplify documents at the sentence level, thereby making the entire system better at the simplification task.

Acknowledgments

Thank you Bernardo for an amazing class. I learned more than I could have hoped, and definitely feel better equipped for further studies into this fascinating field. I look forward to working with you again next term and cannot wait to see what you have in store for me!

[1] For example, the sentence *This is a test .* may be padded to [*This is a test . pad pad pad pad pad*] for a pad length of ten.

[2] In the example above, possible padding variations would include [*pad pad This is a test . pad pad pad*] or [*pad pad pad pad This is a test . pad*].

References

[1] Advaith Siddharthan. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298, 2014.

[2] Yoshua Bengio, Rèjean Ducharme, Pascal Vincent, and Jauvin Christian. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[3] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[4] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413, 2013.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781, 2015.

[9] Sebastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2015.

[10] Tong Wang, Ping Chen, Kevin Amaral, and Jipeng Qiang. An experimental study of lstm encoder-decoder model for text simplification. *arXiv preprint arXiv:1609.03663*, 2016.

[11] Tong Wang, Ping Chen, Kevin Amaral, and Jipeng Qiang. An experimental study of lstm encoder-decoder model for text simplification. September 2016.

[12] Ramesh Nallapati, Bing Xiang, and Bowen Zhou. Text summarization. *arXiv preprint arXiv:1602.06023*, 2016.

[13] Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16*, pages 93–98, 2016.

[14] Kristian Woodsent and Mirella Lapata. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Edinburgh, United Kingdom, July 2011.

[15] David Kauchak. Improving text simplification language modeling using unsimplified text data. *ACL*, 1:1537–1546, 2013.

[16] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

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

[18] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*, 2016.

[19] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*, pages 1480–1489, 2016.

[20] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.