Neural Speed Reading Audited

Anders Søgaard
Dpt. of Computer Science
University of Copenhagen
soegaard@di.ku.dk

Abstract

Several approaches to *neural speed reading* have been presented at major NLP and machine learning conferences in 2017–20; i.e., "human-inspired" recurrent network architectures that learn to "read" text faster by skipping irrelevant words, typically optimizing the joint objective of minimizing classification error rate and FLOPs used at inference time. This paper reflects on the meaningfulness of the speed reading task, showing that (a) better and faster approaches to, say, document classification, already exist, which also learn to ignore part of the input (I give an example with 7% error reduction and a 136x speed-up over the state of the art in neural speed reading); and that (b) any claims that neural speed reading is "human-inspired", are ill-founded.

1 Introduction

A new natural language processing (NLP) task, called *neural speed reading*, or simply *speed reading*, has attracted a lot of attention within the last four years (Yu et al., 2017; Johansen and Socher, 2017; Gui et al., 2017; Huang et al., 2017, 2018; Seo et al., 2018; Fu and Ma, 2018; Yu et al., 2018b; Hansen et al., 2019; Li et al., 2019; Tao et al., 2019; Liu et al., 2020). The basic idea is to model "*human* speed reading techniques" (Fu and Ma, 2018) for more efficient NLP, including document classification, named entity recognition, and machine comprehension. Neural speed reading architectures are typically recurrent neural networks – long short term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) – that jointly learn to process documents and ignore parts of them in making their decisions.

The term "speed reading" comes from psycholinguistics, where it refers to fast-paced human reading, associated with fewer eye fixations, short fixation times, and longer saccades. However, while

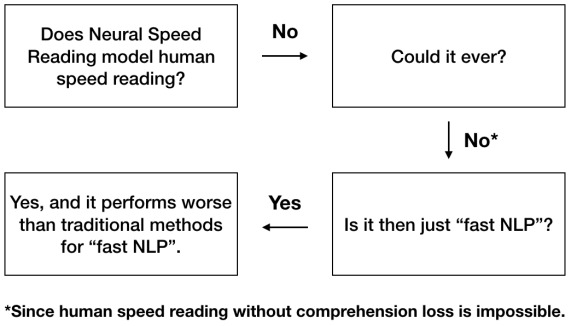

Figure 1: Our argument, schematically

some of the above authors claim to model human speed reading, e.g., Fu and Ma (2018), they do not evaluate their ability to do so, say by evaluating against eye-tracking data from readers.[1] Surveying the psycholinguistics literature, however, it turns out that the notion of "human speed reading" is surrounded by controversy; there is in fact little evidence that humans can read significantly faster without also incurring a significant information loss (McLaughlin, 1969; Rayner et al., 2016).

Neural speed reading is therefore not – and can never be – a cognitive modeling effort of modeling human speed reading strategies. Neural speed reading is therefore *not* a new task, but reduces to the well-known task of computationally efficient NLP, e.g., document classification with a time budget (Xu et al., 2012; Nan et al., 2016; Nan and Saligrama, 2017). Moreover, as I show be-

[1] Such data is readily available for normal-paced reading in the form of corpora such as the Dundee Corpus and the GECO Corpus: https://www2.ling.ohio-state.edu/golddundee/ and http://expsy.ugent.be/downloads/geco/, respectively. These datasets have been used in machine learning experiments aimed at predicting fixations during reading (Nilsson and Nivre, 2009; Matthies and Søgaard, 2013), as well as as auxiliary data for various NLP tasks (Barrett and Søgaard, 2015; Klerke et al., 2016). Klerke et al. (2016), for example, show that jointly predicting fixations during reading is beneficial for a sentence compression model, trying to shorten and simplify input sentences.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 148–153
November 16 - 20, 2020.

low, neural speed reading architectures perform poorly compared to simple baseline approaches to fast document classification.

Contributions In sum, this paper makes the following contributions: (a) I argue speed reading reduces to computationally efficient NLP, e.g., fast document classification. (b) I therefore present a heads-to-heads comparison of a state-of-the-art speed reading architecture to a simple n-gram-based classifier. (c) Our simple n-gram-based classifier is shown to be significantly better and faster than the speed reading architecture.

2 Speed Reading

Speed reading, as a machine learning task, was only introduced about three years ago, but has attracted a lot of attention (Yu et al., 2017; Johansen and Socher, 2017; Huang et al., 2017; Seo et al., 2018; Fu and Ma, 2018; Yu et al., 2018b; Hansen et al., 2019): All proposed models so far are extensions of recurrent neural networks for text classification or sequence labeling - mostly long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) - that learn to either skip, skim or re-read words, jump elsewhere in the text or to make early predictions. As mentioned, none of the papers on neural speed reading, some of which are reviewed below, evaluate the extent to which they simulate *human* speed reading strategies. While the idea of human speed reading has intrigued modern society for decades – at least since Evelyn Wood introduced her Reading Dynamics training program in 1959 – the psycholinguistic literature argues very convincingly that human speed reading is in fact implausible:[2] The reason is physical: In order to read, people need to move their eyes so as to place the fovea[3] over the region that they want to process (Rayner et al., 2016). Fixation times (150-200ms) and saccade times (20-35ms) are relatively fixed, and this puts a lower bound on reading time.[4] In other words, while speed reading courses claim readers can learn to obtain information from a large area of text in a single fixation, it seems there is little scientific support for such claims: Humans can not read significantly faster without a significant loss in comprehension. Speed reading architectures have therefore also not been evaluated against, say, eye-tracking data from human speed reading experiments, and we therefore argue neural speed reading simply reduces to fast NLP. We review prominent architectures below.

Speed reading architectures Yu et al. (2017) present a model that reads a fixed number of words, and then may decide to jump up to n words ahead or stop reading. The number of jumps permitted is also bounded to m, the objective is to learn how best to spend the m jumps. The authors propose to use simply policy gradient training (Williams, 1992) (because jumps lead to non-decomposable loss), using classification accuracy as a reward function. Note that it is *not* part of the objective to minimize the number of FLOPs. They report their modified LSTM with jumping is up to 6 times faster than their baseline LSTM, while maintaining the same or even better accuracy. Extending the work of Yu et al. (2017), Yu et al. (2018b) use actor-critic training rather than policy gradient training and a reward function combining task performance and FLOP reduction. The approach taken in Fu and Ma (2018) is also very similar to that of Yu et al. (2017), except their model allows backwards jumps, enabling re-reading of text snippets. Huang et al. (2017) propose a simple speed reading architecture that simply learns when to stop reading. Seo et al. (2018) combine a large and a small recurrent neural network and learns, at each time step, to choose which to use. The small network is thought of as only *skimming* the text. Since this discrete choice leads to non-decomposable loss, they train the network using Gumbel softmax. Campos et al. (2018) presents an architecture that can learn to *skip* (rather than skim) individual words. Johansen and Socher (2017) introduce a speed reading model for sentiment classification, in which a simple sub-model determines whether or not to use an LSTM or an n-gram-based classifier. Their proposal, however, relies on the assumption that an LSTM, in general, outperforms (all) n-gram-based classifiers on these document classification problems. We show that this assumption is false, and that (some) n-gram-based classifiers consistently outperform state-of-the-art speed reading architectures.

Hansen et al. (2019) will be our baseline in the experiments below. We therefore describe

[2] It "is unlikely that readers will be able to double or triple their reading speeds while still being able to understand the text as well . . . " (Rayner et al., 2016)

[3] The fovea is the 1° region around the center of vision.

[4] Even if a reader has no processing difficulties, suffers from no fatigue effects, and only fixates on every second word, she would at most be able to read 600 words per minute. On average, skilled readers skip 30% of words and regress back to words in 10% of their eye movements (Rayner et al., 2016).

this model in some detail: STRUCTURAL-JUMP-LSTM combines a standard LSTM network with two simple agents: the skip agent and the jump agent. Each of these agents predicts a transition distributions, from which actions can be sampled from: Skipping amounts to ignoring the next word in the sequence, i.e., not updating the LSTM, whereas jumping ignores all information up to some point, which can either be the next clausal separator symbol (, or ;), or the next sentence segmentator (., ! or ?), or the end of the document.[5] The motivation for adding the jump agent, which is what differentiates STRUCTURAL-JUMP-LSTM from previous models, is the computational advantage (FLOP reduction) of being able to ignore n words without having to query the skip agent n times. The input in each time step is the previous actions of the skip agent, of the jump agent, and of the current input. The output from the previous LSTM state representation is used by the agents in combination with the input to make a skip/jump decision – if the word is skipped or jumped over, the LSTM state will not be updated. Both agents consist of a fully connected layer, but which is significantly smaller than the LSTM cell size. Using these agents to skip part of the input reduces the number of FLOPs used when processing input sequences. Hansen et al. (2019) use a combination of maximum likelihood and actor-critic training to train their STRUCTURAL-JUMP-LSTM architecture. They do so in order to jointly minimize classification error and the number of reads. Since the number of reads does not decompose over the input, they cannot rely solely on maximum likelihood training and instead use A3C training (Mnih et al., 2016) with a baseline offset.[6]

3 Experiments

Datasets In our experiments, we use the three document classification datasets most commonly used in the speed reading literature: IMDB and ROTTENTOMATOES are both datasets of positive and negative movie reviews collected from the IMDb moview review database. IMDB is larger than ROTTENTOMATOES and also contains significantly longer documents. AG NEWS, on the other hand, is a document classification dataset, where news are classified by their topic. The AG NEWS corpus consists of news articles from a corpus of news articles on the web, focusing only on the four largest classes. The dataset contains 30,000 training examples for each class, and 1,900 examples for each class for testing. All three datasets are balanced classification tasks, and we thus simply report accuracies on held-out evaluation samples.

On all three datasets, Hansen et al. (2019) report state-of-the-art classification performance (Accuracy) and FLOP reductions (FLOP-r).[7] We therefore use their system as our baseline. We refer to their model as STRUCTURAL-JUMP-LSMT (SJ-LSTM). As we were not able to reproduce results with their code base,[8] we use their reported results for comparison.

Our classifier is a simple multi-layered perceptron with a single hidden layer of 300 dimensions. We use the Scikits implementation with default parameters,[9] except that we use early stopping and set $\beta_1 = 0.95$ based on a held-out (10%) portion of the IMDb training data.[10] For all three datasets, we use the same hyper-parameters, and train our classifier on the k ($k = 6,000$) most frequent n-grams in the training split, with $n \in \{1, 2, 3\}$.[11] We use

[5] The authors do not perform clause and segment segmentation and thus ignore the ambiguity of punctuation symbols; the jump actions therefore only *approximately* jump to the end of the current clause/sentence.

[6] One difference between our n-gram-based classifier and Hansen et al. (2019) is that they optimized several hyper-parameters based on performance on task-specific validation data. We use the same hyper-parameter setting, optimized on IMDB held-out training data, across all tasks to avoids overly optimistic performance estimates. This, in turn, means our improvements over this state-of-the-art architecture for speed reading are even more remarkable.

[7] While their classification performance is state of the art among speed reading architectures, others have reported much better performance on the same datasets. Howard and Ruder (2018) report an accuracy of 0.951 on AG NEWS, which is an error reduction of 56% over the result reported for STRUCTURAL-JUMP-LSTM in Hansen et al. (2019). Tay et al. (2018) present an architecture that is in many ways very similar to state-of-the-art speed reading architectures. It does not skip any words, but for each word queries a controller network that determines what part of the main network to use. They report a classification performance of 0.928 on IMDB, which is an error reduction of 39% over the result reported for STRUCTURAL-JUMP-LSTM in Hansen et al. (2019). Curiously, Yu et al. (2017) also report slightly higher performance than Hansen et al. (2019) on IMDB and AG NEWS, but much worse performance on ROTTENTOMATOES; this seems to be mostly due to differences in their baseline LSTM architectures, though.

[8] https://github.com/Varyn/Neural-Speed-Reading-with-Structural-Jump-LSTM

[9] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. Default parameters: Adam, ReLUs, $b = 200$, β_2=0.999, ϵ=$1e^{-08}$.

[10] We considered $\beta_1 \in \{0.9, 0.95, 0.99\}$.

[11] The values of k and n are also based on a held-out (10%) portion of the IMDb training data. We considered $k \in \{1000, 2000, \ldots, 8000\}$ and restricting n to $\{1, 2\}$,

	IMDB			ROTTENTOMATOES			AG NEWS		
	Acc	ΔAcc	FL-r	Acc	ΔAcc	FL-r	Acc	ΔAcc	FL-r
LSTM	0.882	-	-	0.787	-	-	0.880	-	-
Hansen et al. (2019)	0.882	0.000	6.3x	0.790	0.003	2.1x	0.883	0.003	2.4x
Seo et al. (2018)	-	0.001	5.8x	-	0.016	2.1x	-	0.001	1.4x
Yu et al. (2018b)	-	0.005	3.4x	-	0.002	1.5x	-	0.001	1.7x
Fu and Ma (2018)	-	0.008	2.1x	-	0.007	1.7x	-	0.020	1.3x
SIMPLE MLP	**0.886**	0.004	1,101x	**0.791**	0.004	29x	**0.903**	0.023	335x

Table 1: Comparing the performance of our simple n-gram-based classifier (SIMPLE MLP) with state-of-the-art speed reading models. FLOP reductions (FLOP-r) are relative to the LSTM baseline architecture in Hansen et al. (2019). The average error reduction over STRUCTURAL-JUMP-LSTM is 7%, and the average speed-up over STRUCTURAL-JUMP-LSTM is 136x.

no preprocessing beyond lower-casing.

We report accuracies in Table 1. We also report the absolute improvement (ΔAcc) and FLOP reductions (FL-r) over an LSTM baseline, following Hansen et al. (2019). The FLOP reductions are computed by dividing the FLOPs used by the baseline architecture at test time by the number of FLOPs used by our systems at test time: Our first observation is that our n-gram-based classifier consistently outperforms the reported performance of the STRUCTURAL-JUMP-LSTM architecture. This is remarkable, since the STRUCTURAL-JUMP-LSTM is a novel deep learning architecture, which employs more parameters and takes considerably less time to train: Our total training time corresponds roughly to training the baseline LSTM architecture for one epoch, but the n-gram-based architecture is significantly faster at inference time, as measured in FLOP reductions. On average, we reduce 136 times as many FLOPs as STRUCTURAL-JUMP-LSTM. The n-gram-based classifier is also easier to parallelize than the STRUCTURAL-JUMP-LSTM. The n-gram-based classifier's accuracies – both relative and absolute – are slightly better for IMDB and ROTTENTOMATOES, and considerably better for AG NEWS.

4 Discussion and conclusion

We have argued that traditional n-gram classifiers are fully adequate baselines for neural speed reading architectures, in the context of document classification. In some of the neural speed reading papers cited above, including Hansen et al. (2019), the authors also report results on sequence labeling problems such as entity recognition or machine comprehension. Are those experiments more meaningful than the document classification experiments? Not really. Yu et al. (2018a), for example, present a non-recurrent machine comprehension model based on local convolutions and attention that is 4–9x faster at inference time than their recurrent baseline model and achieves significantly superior performance. Wu et al. (2017) present an even simpler non-recurrent model based only on convolutions that is 100x faster than their recurrent baseline model and achieves the same performance. Both papers are good examples of significantly faster reading strategies for a sequence labeling task – in this case, machine comprehension – that seem to outperform neural speed reading architectures by some margin. For a more direct comparison, Trischler et al. (2016) show that using only convolutional encoders and similar scores on the CBT-CN dataset seem to outperform their own LSTM baseline, as well as STRUCTURAL-JUMP-LSTM, by some margin.[12]

In conclusion, we presented a comparison of neural speed reading architectures with a simple n-gram-based classifier, and showed how this classifier is superior to all proposed neural speed reading architectures on standard document classification tasks used to benchmark neural speed reading architectures, both in terms of performance (7% error reduction) and speed (136x reduction in FLOP). Citing research in psycholinguistics, we observed that speed reading without comprehension loss cannot be observed in humans, and for this reason, we argue that the task of neural speed reading has been a digression, and that researchers should instead focus on simply building fast NLP models.

$\{1, 2, 3\}$, and $\{1, 2, 3, 4\}$.

[12] The baseline LSTM in Hansen et al. (2019) scores 0.045 lower than the LSTM baseline in Trischler et al. (2016) on the CBT-CN dataset.

References

Maria Barrett and Anders Søgaard. 2015. Reading behavior predicts syntactic categories. In *CoNLL*.

Victor Campos, Brendan Jou, Xavier Giro i Nieto, Jordi Torres, and Shih-Fu Chan. 2018. Skip RNN: Learning to skip state updates in recurrent neural networks. In *ICLR*.

Tsu-Jui Fu and Wei-Yun Ma. 2018. Speed reading: Learning to read forbackward via shuttle. In *EMNLP*.

Tao Gui, Qi Zhang, Lujun Zhao, Yaosong Lin, Minlong Peng, Jingjing Gong, and Xuanjing Huang. 2017. Long short-term memory with dynamic skip connections. *Computing Research Repository*, arXiv:1811.03873.

Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. 2019. Neural speed reading with Structural-Jump-LSTM. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *ACL*.

Ting Huang, Gehui Shen, and Zhi-Hong Deng. 2018. Leap-lstm: Enhancing long short-term memory for text categorization. *Computing Research Repository*, arXiv:1905.11558.

Zhengjie Huang, Zi Ye, Shuangyin Li, and Rong Pan. 2017. Length adaptive recurrent model for text classification. In *ACM Conference on Information and Knowledge Management*.

Alexander Johansen and Richard Socher. 2017. Learning when to skim and when to read. In *Rep4NLP*.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *NAACL*.

Zhe Li, Peisong Wang, Hanqing Lu, and Jian Cheng. 2019. Reading selectively via binary input gated recurrent unit. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5074–5080. International Joint Conferences on Artificial Intelligence Organization.

Xianggen Liu, Lili Mou, Haotian Cui, Zhengdong Lu, and Sen Song. 2020. Finding decision jumps in text classification. *Neurocomputing*, 371:177 – 187.

Franz Matthies and Anders Søgaard. 2013. With blinkers on: Robust prediction of eye movements across re aders. In *EMNLP*, Seattle, Washington, USA.

G. H. McLaughlin. 1969. Reading at "impossible" speeds. *Journal of Reading*, 12:449–454.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*.

Feng Nan and Venkatesh Saligrama. 2017. Adaptive Classification for Prediction Under a Budget. In *NeurIPS*.

Feng Nan, Joseph Wang, and Venkatesh Saligrama. 2016. Pruning Random Forests for Prediction on a Budget. In *NeurIPS*.

Matthias Nilsson and Joakim Nivre. 2009. Learning where to look: Modeling eye movements in reading. In *CoNLL*.

Keith Rayner, Elizabeth Schotter, Michael Masson, Mary Potter, and Rebecca Treiman. 2016. So Much to Read, So Little Time: How Do We Read, and Can Speed Reading Help? *Psychological Science in the Public Interest*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirz. 2018. Neural speed reading via Skim-RNN. In *ICLR*.

Jin Tao, Urmish Thakker, Ganesh Dasika, and Jesse Beu. 2019. Skipping rnn state updates without retraining the original model. In *Proceedings of the 1st Workshop on Machine Learning on Edge in Sensor Systems*, SenSys-ML 2019, page 31–36, New York, NY, USA. Association for Computing Machinery.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Recurrently Controlled Recurrent Networks. In *NeurIPS*.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural language comprehension with the EpiReader. In *EMNLP*, Canberra, Australia.

Ronald Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Felix Wu, Ni Lao, John Blitzer, Guandao Yang, and Kilian Weinberger. 2017. Fast reading comprehension with convnets. *Computing Research Repository*, arXiv:1711.04352.

Zhixiang Xu, Kilian Weinberger, and Olivier Chapelle. 2012. The Greedy Miser: Learning under Test-time Budgets. In *NeurIPS*.

Adams Wei Yu, Hongrae Lee, , and Quoc Le. 2017. Learning to skim text. In *ACL*.

Adams Weis Yu, David Dohan, Minh-Thang Luong, Ruy Zhai, Kai Chen, Mohammad Norouzi, and Quoc Le. 2018a. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *ICLR*.

Keyi Yu, Yang Liu, Alexander Schwing, , and Jian Peng. 2018b. Fast and accurate text classification: Skimming, rereading and early stopping. In *ICLR*.