Effificent Language Modeling of Long-Term Dependencies

Stanford CS224N Custom Project **[OPTION 3: GRADED]**

Manan Shah
Department of Computer Science
Stanford University
manans@stanford.edu

Abstract

The transformer architecture [1] has received significant research attention due to its ability to effectively model word-level and global contextual dependencies exclusively using self-attention, thereby enabling language models such as BERT [2] to achieve state-of-the-art performance. However, the quadratic cost of attention and linear memory cost per layer have severely limited the transformer to operate solely on short sequences, a problem partially addressed by the reversible layers and LSH attention proposed in the reformer [3]. In this work, we study and improve properties of the reformer as a language model, introducing k-means clustering for attention and connection tying in reversible layers to improve reformer complexity and representational power. We evaluate our model on masked language modeling as well as selected GLUE benchmarks, and we find that our modifications significantly improve training times and model capacity. The methods presented in our work are generalizable to other attention-based models and have the potential to drastically improve the efficacy of long-term language dependency modeling.

Key Information: This is a custom project with mentor Hang Jiang and no external collaborators; it is not shared with another course.

1 Introduction

The transformer architecture, first introduced in [1], dramatically disrupted the field of computational language modeling due to its sole use of self-attention as mechanism to effectively learn representations from natural language without many of the drawbacks present in recurrent networks. In particular, while recurrent models suffer from vanishing gradients in backpropagation and slow training due to sequential processing of input sequences, transformers eschew the idea of recurrence, instead learning global dependencies in sequences through attention mechanisms. As a result of this unique approach for sequence modeling, transformers have been able to achieve remarkable parallelism and generalizability, leading to their widespread adoption in language modeling frameworks. In particular, BERT [2], a natural language model developed by Google in 2019, utilized a series of stacked transformers in a bidirectional encoder-only framework designed with two tasks focused on learning word and sequence representations. The conceptually simple nature of BERT alongside its empirical success in obtaining state-of-the-art results on eleven GLUE [4] natural language processing tasks cemented the importance of self-attention and the transformer architecture in the field of natural language processing.

However, in spite of their success, transformers suffer from two significant limitations: (1) computing attention on sequences of length n is an $O(n^2)$ operation, and (2) the memory of a model increases linearly in its number of layers. These issues have limited transformer-dependent language models such as BERT to operate on sequences of maximum 512 tokens, severely limiting their representational power and increasing model training times. Recently, the reformer [3] proposed modifying attention with locality-sensitive hashing to address (1) and introducing reversible layers that perform

backpropagation without storing intermediate layer activations to address (2). Implementing these adjustments allowed the reformer to generalize attention to thousands of tokens in an efficient manner, thereby paving the way for larger and more efficient attention-based language models.

In this work, we aim to extend and improve properties of the reformer as part of a language model. In particular, we (a) introduce k-means attention and reversible layer connection tying to enhance the expressivity of the reformer while reducing its training time complexity, (b) conduct empirical evaluation of the (original and modified) reformer's performance as a building block in a masked language modeling framework, and (c) perform empirical evaluation of our model on masked language modeling as well as downstream predictions on a variety of GLUE tasks. We implement our adjustments as augmentations to an existing PyTorch reformer implementation [5]; code is located at `https://github.com/mananshah99/reformer-pytorch`.

2 Related Work

Since the conception of the transformer architecture in 2017, significant progress has been made to overcome the quadratic cost of attention and generalize the attention mechanism to significantly longer sequences. Much of this work has attempted to leverage the sparsity of weights in attention layers (as in [6], [7], and [8]); in particular, [6] utilized sparse factorizations of the attention matrix to reduce computational complexity from $O(n^2)$ to $O(n\sqrt{n})$. This method was further generalized by [9], which learned context-dependent sparsity patterns in attention matrices to improve interpretability. Promising work has additionally been conducted on reducing the memory cost of backpropagation due to intermediate layer activation storing; taking cues from normalizing flow networks, [10] proposed reversible residual networks that perform backward passes with constant memory complexity. Along with attention-specific modifications, engineering optimizations such as gradient checkpointing and model quantization [11] have shown promise in facilitating the training of deeper transformers with larger context lengths, although these techniques remain limited in their scope and potential benefits beyond thousand-token sequence lengths.

The reformer, leveraging developed intuition that the majority of tokens computed with dense attention carry little weight, developed a hash-based attention mechanism further reducing attention complexity from $O(n\sqrt{n})$ to $O(n \log n)$. It additionally replaced transformer residual connections with reversible connections as in [10], obtaining $O(1)$ memory complexity for additional layers and allowing for more efficient language modeling of long-term dependencies. As a result, developing augmentations to improve the speed and representational power of the reformer holds significant promise for the advancement of sequence learning as a whole.

3 Approach

In this section, we detail our approach to improving the performance and representational power of the reformer. We begin with an overview of the core attention and feedforward mechanisms exhibited in the transformer and their adaptations in the reformer. We next focus on (1) our replacement of LSH hash-bucketing with learned k-means to improve training times and model performance, and (2) our introduction of a connection-tying term as part of the reversible layer framework to increase model capacity and generalizability. As the reformer further reduces transformer memory complexity by setting the query and key projections to be identical (so that $Q = K$), we adopt this notation where necessary for the remainder of this work.

3.1 Background

In the transformer, attention is computed by first setting $Q = XW_Q, K = XW_K, V = XW_V$ where all W matrices are projection matrices from the input space $X \in \mathbf{R}^{l \times d}$ for sequence of maximum length l. Attention, the only mechanism for learning contextual representations, is then computed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Multi-head attention in [1] is simply a concatenation of multiple individual attention heads with a linear projection to ensure dimensions are preserved. After computation of attention, outputs are

passed through a feed-forward layer composed of two linear layers and a rectified linear unit; in particular, we have that

$$\text{FeedForward}(X) = \text{ReLU}(XW_1 + b_1)W_2 + b_2 \tag{2}$$

Additionally, in the transformer architecture, both attention and feedforward layers are followed by layer normalization and residual connections. As evinced by (1), the matrix product requires quadratic time to compute, and as shown in (2) the memory required to store the activations in each layer is linear in the number of layers of the transformer. The reformer attempts to resolve these prohibitive constraints by introducing locality-sensitive hashing (LSH) attention and reversible layers.

LSH attention was primarily motivated by the recognition that the computation of attention is heavily biased towards similar vectors (as their dot products are the largest). As a result, identifying similar vectors and only computing attention between those vectors is a reasonable approximation to overcoming the quadratic complexity of attention; doing so with LSH ensures sub-quadratic identification of such vectors. In particular, the bucketing stage of LSH attention is performed by fixing a random rotation matrix R and defining hash function $h(x) = \text{argmax}([xR; -xR])$; doing so conforms to a known LSH scheme proposed in [12]. In order to minimize the probability of similar items hashing to different buckets in this mechanism, the reformer performs multi-round LSH attention with different hash functions (each associated with distinct rotation matrices). LSH attention is subsequently performed in parallel for each round, and final buckets are obtained by computing unions over all hashes. The attention mechanism used in the reformer subsequently computes attention elementwise between vectors in the same hash buckets; the reader is referred to Equations (3), (4), and (5) in [3] for further details.

Reversible layers, inspired by [10], were introduced in the reformer to overcome the linear memory cost of (2). Whereas a normal residual layer performs a mapping $x \mapsto y$, a reversible layer maps $(x_1, y_1) \mapsto (y_1, y_2)$ so that the forward pass computes

$$y_1 = x_1 + F(x_2) \qquad \text{and} \qquad y_2 = x_2 + G(y_1)$$

which allows for layers to be reversed by subtracting the residuals. The reformer sets $F =$ Attention (1) and $G =$ FeedForward (2), with layer normalization moved inside the residual blocks. This computation eliminates the requirement of storing layer-wise activations; backpropagation is instead performed according to Algorithm 1 in [10] by computing total derivatives in the reversed step.

3.2 Learned k-Means Attention

Our first proposed augmentation to the reformer focused on improving the expensive computation of angular locality sensitive hashing across the sequence of queries/keys to obtain buckets for each vector. In particular, while the hash-bucketing performed in the original reformer (detailed in Section 3.1) enjoys the theoretical guarantees of a locality-sensitive hashing schema, it requires repeated hashing to reduce error probabilities below a desirable threshold. Furthermore, storing the rotation matrix R and computing rotations for each vector is expensive; such costs may significantly slow training times as well.

In order to improve upon this notion of hash-bucketing, we note that the hash buckets produced in LSH hashing share similarities with clusters in key/query vector space. In particular, clusters in this space perform analogous functionalities to buckets produced by angular hashing; while vectors in the same cluster are similar in a suitable Euclidean norm, vectors in the same bucket are similar componentwise. Taking advantage of these analogous concepts, we propose reducing the memory complexity of hashing into n buckets by learning n k-means centroids within the set of L2-normalized query/key vectors; normalization is essential to prevent increasingly large dot products. We further randomly re-initialize m different clusters (one for each random rehash performed in LSH attention) to reduce the probability of suboptimal starting points [13].

Our implementation stores a tensor of per-cluster means associated with the input set of key/query vectors; at each training iteration, we compute clusters for each vector and reassign means according to the canonical k-means approach. We assign final clusters according to the mode of the predictions across all n random re-initializations. This iterative k-means approach allows us to provide stronger guarantees on bucketing (an approximation ratio of $O(\log n)$ at maximum) in a learned manner as opposed to the more time and memory-consuming operation of repeated random rehashing.

3.3 Reversible Layer Tied Connections

Our second proposed improvement to the Reformer focused on improving the representational power of the reversible residual networks. The canonical reversible layer used in the Reformer performs a forward pass on pairs of inputs/outputs $(x_1, x_2) \mapsto (y_1, y_2)$ as in the black portion of Equation (3); our modifications are highlighted in red.

$$y_1 = x_1 + F(x_2) + H(x_1, x_2) \qquad \text{and} \qquad y_2 = x_2 + G(y_1) \tag{3}$$

In particular, the addition of a "connection tying" term $H(x_1, x_2)$ allows for operations more complex than simple summations to learn connections between the vectors, thereby increasing the representational complexity of the reversible networks. In accordance with Equation (4), we additionally adjusted the reversible residual backpropagation algorithm to express the total derivatives of x_1 and x_2 (represented as $\overline{x}_1$ and $\overline{x}_2$ respectively) and the weight updates of H as

$$\overline{x}_2 = \overline{y}_2 + \left(\left(\frac{\partial F}{\partial x_2} \right)^T + \left(\frac{\partial H}{\partial x_2} \right)^T \right) \left(\overline{y}_1 + \left(\frac{\partial G}{\partial y_1} \right) \overline{y}_2 \right) \tag{4}$$

$$\overline{x}_1 = \left(\overline{y}_1 + \left(\frac{\partial G}{\partial y_1} \right) \overline{y}_2 \right) + \left(\frac{\partial H}{\partial x_1} \right)^T \overline{y}_1 \tag{5}$$

$$\overline{w}_H = \left(\frac{\partial H}{\partial w_H} \right)^T \overline{y}_1 \tag{6}$$

where $\overline{y}_1$ and $\overline{y}_2$ represent the total derivatives of the activations y_1 and y_2 which are provided as input in the backwards pass. The reader is referred to Algorithm 1 in [10] for a detailed listing of notation as well as the remainder of the backward pass algorithm. For our particular augmentation to the reformer, we follow [3] in setting F as an attention layer and G as a feedforward layer; we additionally set H to be a feedforward layer operating along the concatenation of $[x_1, x_2]$ and producing output of dimension equal to $dim(x_1) = dim(x_2)$. Our implementation of tied connections within reversible layers is an augmentation of the existing reversible residual network framework outlined in [10].

3.4 Experimental Baselines

In order to understand the efficacy of learned k-means attention and reversible layer tied connections on language modeling with the reformer, we constructed a simple language model around the reformer framework. In particular, our BERT-inspired model consisted of a fixed position embedding followed by a feed-forward layer and a series of reformer blocks (each of which contained attention, feed-forward, and residual connections).

As we lacked sufficient time to train our language model for weeks and compare results with BERT state-of-the-art performance, we defined our first baseline to be the performance of our constructed language model using full shared key-query attention and reversible residual layers without connection tying. We additionally defined a second baseline consisting of our language model using LSH attention (as in [3]) and traditional reversible residual layers. We evaluated our chosen baselines against our fully augmented framework along with ablations of k-means attention and reversible residual layers with tied connections.

4 Experiments

In this section, we present results of our two aforementioned baselines (Section 3.4), full augmented model, and its ablations on a masked language modeling task as well as several selected GLUE benchmarks; we emphasize performance as well as training and inference times across all models. All experiments were conducted on a single NVIDIA Tesla M60 GPU with 8GB of memory.

4.1 Masked Language Modeling

Dataset and Experimental Details. In order to evaluate the efficacy of our models at learning representations of natural language, we utilized a masked language modeling task where 80% of tokens were masked, 10% were randomly chosen, and 10% were unchanged (using a pre-trained BERT tokenizer [14]). For this task, we selected a massive dataset of 3,036 English books written by

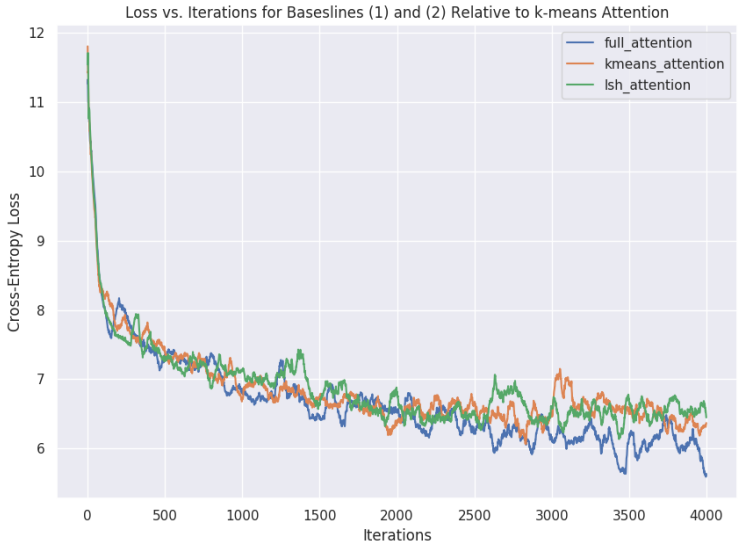

Figure 1: Pre-training loss for baseline (1) and (2) relative to our k-means attention variant. Evaluation loss and perplexities are reported in Section 4.1.

142 authors from Project Gutenberg [15]. We utilized code from [5] to perform masking and padding, and we split our dataset into 90% train, 10% test. All experiments were conducted with a maximum sequence length of 512, reformer uniform hidden layer dimension of 512, depth of 6, and 8 attention heads; the only variant was the ablated variable. Training was conducted for 5K iterations (batch size 32) with an adaptive learning rate schedule as in [16]. Evaluation was performed with respect to the cross-entropy loss and perplexity [17] metrics.

Results. Figure 1 represents training loss versus iteration number for both baselines as well as the k-means attention model after 5K iterations. Our results upon evaluation on a held-out test set are represented in Table 1, where metrics were computed by masking arbitrary words from unseen books and measuring the models' ability to predict them.

Model Type	Evaluation Loss	Evaluation Perplexity
Baseline 1 (Full Attention)	5.58	968.87
Baseline 2 (LSH Attention)	4.86	1054.37
k-means Attention	4.90	726.48

Table 1: Evaluation results for baselines and our proposed models on masked language modeling.

Furthermore, Figure 2 represents average training times observed for both k-means and the LSH attention baseline, with significant improvements using the k-means approach as hypothesized. Our findings therefore point to the efficacy of k-means attention as a significant boost not only performance-wise but also complexity-wise in the reformer architecture.

We additonally performed experiments with reversible layer tied connections, but those showed diverging losses after 1.2K iterations; debugging has been slowed due to the exceptional circumstances surrounding COVID-19. As a result, we have not reported our results for these findings, although the code and associated scaffolding exists in the indicated repository for the reader to examine and test on their own datasets.

4.2 GLUE Evaluation

Dataset and Experimental Details. In order to evaluate our models' abilities to understand natural language beyond predicting masked words, we additionally conducted evaluation along numerous

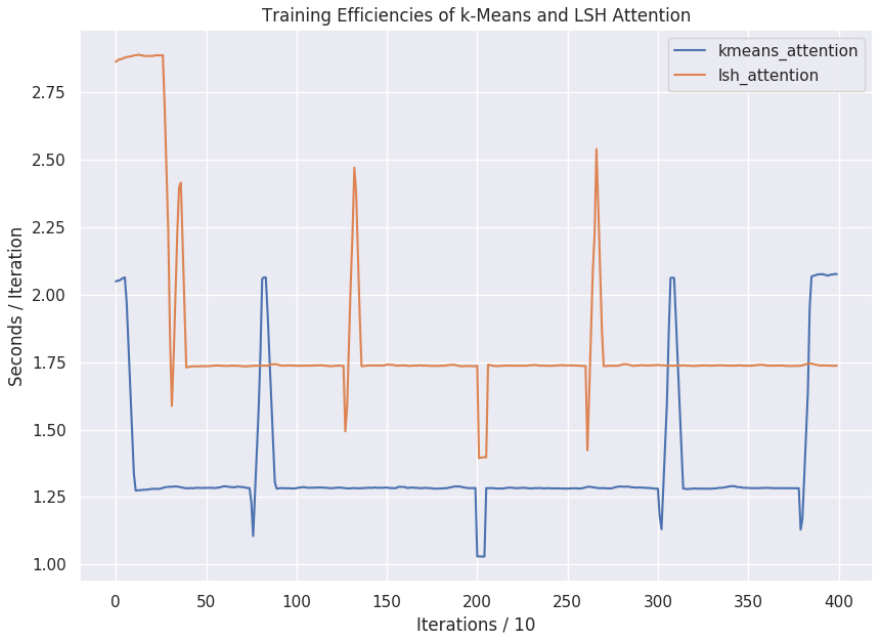

Figure 2: Seconds per iteration for the LSH attention baseline and k-means attention during pre-training. Note the significant benefits of our proposed k-means attention in improving training times.

GLUE verticals [4]. In particular, we focused on performing evaluation along the CoLA and SST-B datasets; the former is an acceptability task, while the latter is focused on sentence similarity. We utilized the canonical train/test splits for both datasets, so that the CoLA dataset contained 10,000 training examples and 1,100 test examples while the SST-B dataset contained 7,000 training examples and 1,400 test examples.

Following the canonical pretraining/finetuning pipeline for BERT-like language models [2], we approached these tasks by utilizing our pretrained models on the masked language modeling task from Section 4.1. In particular, we added a classification head over all output embeddings from our reformer language models and finetuned each model on the specified GLUE tasks. GLUE evaluation was aided by scripts from the HuggingFace team Transformers library [14].

Results. Our results for both of these tasks were unfortunately hampered by the limited size of our pretraining dataset, the fact that we only pretrained for 5K iterations, and the small size of our model. Indeed, we obtained near-zero Matthew's correlation coefficient on the CoLA dataset, and a near-zero Pearson correlation coefficient on the STS-B dataset. We hope to pre-train for longer periods of time with a deeper model in the future in order to effectively understand the impact of our improvements on downstream linguistic tasks.

5 Analysis and Future Work

Performing evaluation the masked language modeling task (Section 4.1) and datasets from the GLUE benchmark (Section 4.2) helped verify the value of our proposed augmentations to the reformer framework along both performance and optimization verticals. In particular, our results from the masked language modeling task indicated that the use of k-means attention alone reduced language model training times by 30%. Furthermore, the use of learned k-means attention was able to increase the representational power of the reformer language model to some degree, resulting in reduced evaluation loss and perplexity scores as in Table 1. While we were unable to evaluate the performance improvement of increasing model capacity via tied weights (due to loss divergence or perhaps an error in our backwards pass implementation), we believe our implementation has great promise and we hope to perform future testing in this regard.

After pretraining as above to ensure that our models learned meaningful representations of natural language, we aimed to use finetuning as a mechanism to evaluate the importance of our adjustments in downstream task evaluation. While our limited model size and pretraining dataset yielded minimal results in this regard, we have developed all scaffolding and code needed to pretrain and finetune our augmented reformer models on larger datasets for longer periods of time. We hope to utilize our code and preliminary findings to delve deeper into this area post-COVID19.

6 Conclusion

In this work, we develop, implement, and analyze the two novel mechanisms to reduce the training time and increase the representational power of the reformer architecture as part of a language model. In particular, our conceptualization of k-means attention performs significantly better other attention mechanisms while better identifying vectors that contribute meaningfully to attention scores. Furthermore, our inclusion of a connection-tying layer has the potential to increase model capacity while preserving the per-layer constant memory requirement, although its impact on model performance has yet to be identified due to diverging losses obtained during experimental analysis. Although our work was cut short by increasing tension surrounding COVID-19 as well as limited training times, we hope that our initial results as well as the scaffolding and pipelines we've developed for analyzing our adjustments will prove useful for future work.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[4] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[5] Phil Wang. reformer-pytorch. `https://github.com/lucidrains/reformer-pytorch`, 2020.

[6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[7] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[8] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.

[9] Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

[10] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in neural information processing systems*, pages 2214–2224, 2017.

[11] Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. Fully quantized transformer for improved translation. *arXiv preprint arXiv:1910.10485*, 2019.

[12] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in neural information processing systems*, pages 1225–1233, 2015.

[13] Sébastien Bubeck, Marina Meilă, and Ulrike von Luxburg. How the initialization affects the stability of the -means algorithm. *ESAIM: Probability and Statistics*, 16:436–452, 2012.

[14] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[15] Shibamouli Lahiri. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[16] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*, 2018.

[17] Thomas Gottron and Christian Gottron. Perplexity of index models over evolving linked data. In *European Semantic Web Conference*, pages 161–175. Springer, 2014.