Peng, Hao

Address		**Email** hapeng@cs.washington.edu (update 2021/12/01)	
1207 Westlake Ave N S506 Seattle, WA 98109		**Home Phone** **Cell Phone** (206) 8238085 **Office Phone**	
Current Institution		**Department**	
Location	185 E Stevens Way NE, Office 394, Seattle, WA 98195		
Highest Degree	Ph.D.	**Institution** University of Washington, Paul G. Allen School for Computer Science & Engineerin	**Date** 2022/06 exp
Thesis Advisor	Noah A. Smith		
Research Interests	**Primary** Natural Language Processing/Computational Linguistics		
Secondary	Machine Learning; AI		
Discipline(s)			
Position(s) applied	FACITHACA (complete) FACNYCTECH (complete)		
1. Noah A. Smith, University of Washington; Allen Institute for Artifitial Intelligence, nasmith@cs.washington.edu (teaching) (2021/12/01)			file (PDF, PDF, 2021/12/01)
2. Luke Zettlemoyer, University of Washington; Facebook AI Research, lsz@cs.washington.edu (2021/12/01)			file (PDF, PDF, 2021/12/02)
3. Chris Dyer, DeepMind; Carnegie Mellon University, Language Technologies Institute, cdyer@google.com (2021/12/01)			
4. Yejin Choi, University of Washington; Allen Institute for Artifitial Intelligence, yejin@cs.washington.edu (2021/12/01)			file (PDF, PDF, 2021/12/05)
Faculty contacts:			
US Citizen: No, **Visa Type:** F1			
Pub1: Random Feature Attention. Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. In Proceedings of the International Conference on Learning Representations (ICLR), 2021. **Pub2:** Backpropagating through Structured Argmax using a SPIGOT. Hao Peng, Sam Thomson, and Noah A. Smith. In Proceedings of the Association for Computational Linguistics (ACL), 2018.			
Understand1: I Understand			
Received Materials (same file submitted to multiple jobs is listed under 'common' only)	*common*	*Curriculum Vitae:* file (PDF, PDF, 2021/12/01) *Research Statement:* file (PDF, PDF, 2021/12/01) *Teaching Statement:* file (PDF, PDF, 2021/12/01) *Statement of Contribution to Diversity, Equity, and Inclusion:* file (PDF, PDF, 2021/12/01) *PDF of Most Significant Paper:* file (PDF, PDF, 2021/12/01) *PDF of Other Most Significant Paper:* file (PDF, PDF, 2021/12/01)	
	FACITHACA	*Cover Letter:* file (PDF, PDF, 2021/12/02)	
	FACNYCTECH	*Cover Letter:* file (PDF, PDF, 2021/12/01)	

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Hao Peng
University of Washington
185 E Stevens Way NE
Seattle, WA 98195
+1 (206) 823-8085
hapeng@cs.washington.edu
homes.cs.washington.edu/~hapeng/

Faculty Recruiting Committee
Cornell Tech
Cornell University

December 1, 2021

Dear Members of the Faculty Hiring Committee,

I am writing to apply for the Assistant Professor position at Cornell Tech, recently advertised on your website. I am currently a doctoral candidate in the Paul G. Allen School of Computer Science & Engineering at the University of Washington, supported by a Google Ph.D. Fellowship. I expect to complete my degree by June 2022.

My research primarily focuses on **natural language processing** (NLP) and **machine learning**. Specifically, I design efficient algorithms for state-of-the-art NLP to mitigate its rising computational cost and negative environmental impact. In addition, I build machine learning models that generalize better and are more robust, drawing inspiration from linguistic meaning representations of natural language. This is complemented by my efforts in understanding deep learning architectures using formal analysis and designing more interpretable NLP models. My works have been presented at top-tier NLP and machine learning venues, receiving a best paper nomination at ACL 2018, and featured at ICLR 2021 spotlight.

During my Ph.D. training, I have served as a teaching assistant for graduate-level NLP courses. I am fortunate enough to have mentored six students from diverse backgrounds. Led by the students, the projects are submitted/published at top venues. These experiences confirm that I am an effective advisor, and through them, I find teaching and mentoring rewarding and enjoyable.

My long-term goal is to develop artificial intelligence with general linguistic capabilities. I believe that Cornell Tech would be an ideal place for me to pursue this goal. Enclosed in this application are my CV, selected publications, and documents detailing my research agenda, teaching and mentoring experience, and diversity statement. Please contact me should you need any additional material. I have great respect for your institution and am excited about the prospect of becoming part of it.

Sincerely,

Hao Peng

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Hao Peng
University of Washington
185 E Stevens Way NE
Seattle, WA 98195
+1 (206) 823-8085
hapeng@cs.washington.edu
homes.cs.washington.edu/~hapeng/

Faculty Recruiting Committee
Department of Computer Science
Cornell Bowers CIS College of Computing and Information Science
Cornell University

December 2, 2021

Dear Members of the Faculty Hiring Committee,

I am writing to apply for the Assistant Professor position in the Department of Computer Science at Cornell University, recently advertised on your website. I am currently a doctoral candidate in the Paul G. Allen School of Computer Science & Engineering at the University of Washington, supported by a Google Ph.D. Fellowship. I expect to complete my degree by June 2022.

My research primarily focuses on **natural language processing** (NLP) and **machine learning**. Specifically, I design efficient algorithms for state-of-the-art NLP to mitigate its rising computational cost and negative environmental impact. In addition, I build machine learning models that generalize better and are more robust, drawing inspiration from linguistic meaning representations of natural language. This is complemented by my efforts in understanding deep learning architectures using formal analysis and designing more interpretable NLP models. My works have been presented at top-tier NLP and machine learning venues, receiving a best paper nomination at ACL 2018, and featured at ICLR 2021 spotlight.

During my Ph.D. training, I have served as a teaching assistant for graduate-level NLP courses. I am fortunate enough to have mentored six students from diverse backgrounds. Led by the students, the projects are submitted/published at top venues. These experiences confirm that I am an effective advisor, and through them, I find teaching and mentoring rewarding and enjoyable.

My long-term goal is to develop artificial intelligence with general linguistic capabilities. I believe that the Department of Computer Science at Cornell would be an ideal place for me to pursue this goal. Enclosed in this application are my CV, selected publications, and documents detailing my research agenda, teaching and mentoring experience, and diversity statement. Please contact me should you need any additional material. I have great respect for your institution and am excited about the prospect of becoming part of it.

Sincerely,

Hao Peng

Hao Peng

Contact Information

Paul G. Allen School of CSE
University of Washington
185 Stevens Way
Seattle, WA 98195, USA

hapeng@cs.washington.edu
+1 (206) 823-8085
`https://homes.cs.washington.edu/~hapeng/`

Education

University of Washington, Seattle, WA — 2016 - *present*
Ph.D. in Computer Science and Engineering
Advisor: Noah A. Smith

University of Washington, Seattle, WA — 2016 - 2018
M.S. in Computer Science and Engineering
Advisor: Noah A. Smith

Peking Univeristy, Beijing, China — 2012 - 2016
B.S. in Computer Science, *Summa Cum Laude*

Research Interests

Natural Language Processing, Computational Linguistics, Machine Learning

Research Experience

Research Assistant with Noah A. Smith — 09/2016 - *present*
University of Washington, Seattle, USA

Research Intern with Lingpeng Kong and Dani Yogatama — 06/2020 - 12/2020
DeepMind, UK

Research Intern with Dipanjan Das — 10/2018 - 12/2018
Google Seattle, USA

Research Intern with Dipanjan Das — 06/2018 - 09/2018
Google New York, USA

Research Intern with Chin-Yew Lin — 10/2015 - 06/2016
Microsoft Research Asia, Beijing, China

Research Asistant with Charles Sutton — 07/2015 - 09/2015
University of Edinburgh, Edinburgh, UK

Research Assistant with Zhi Jin — 07/2014 - 06/2015
Peking University, Beijing, China

Honors and Awards

Google Ph.D. Fellowship, 2019

Honorable Mention for Best Paper at ACL, 2018

Jeff Dean - Heidi Hopper Endowed Regental Fellowship, UW, 2016

Research Excellence Award, PKU, 2015

Foundation Fellowship, PKU, 2015

May the Fourth Fellowship, PKU, 2014

Selected Talks

ABC: Attention with Bounded-memory Control
– Invited talk at the University of Hongkong
Virtual. July, 2021

Random Feature Attention
– Invited talk at Peking University
Virtual. May, 2021
– ICLR conference
Virtual. May, 2021
– Invited talk at DeepMind machine translation reading group
Virtual. March, 2021
– Invited talk at DeepMind
Virtual. December, 2020

Rational Recurrences
– Invited talk at the University of Alberta
Virtual. June, 2020
– Invited talk at Peking University
Beijing, China. December, 2018
– EMNLP conference
Brussels, Belgium. October, 2018

A Mixture of $h-1$ Heads is Better than h Heads
– ACL conference
Virtual. July, 2020

Text Generation with Exemplar-based Adaptive Decoding
– NAACL conference
Minneapolis. June, 2019

Backpropagating through Structured Argmax using a SPIGOT
– ACL conference
Melbourne, Australia. July, 2018

Learning Joint Semantic Parsers from Disjoint Data
– NAACL conference
New Orleans. June, 2018

Deep Multitask Learning for Semantic Dependency Parsing
– Invited talk at New York University Shanghai
Shanghai, China. December, 2017

Mentoring

Ivy Guo – B.S., UW CSE	06/2021 - *present*
Ivan Montero – M.S., UW CSE	09/2021 - *present*
Tobias Rhode – M.S., UW CSE	09/2021 - *present*
Zhaofeng Wu – M.S., UW CSE	02/2020 - 06/2021
Michael Zhang – B.S., UW CSE	10/2018 - 05/2019

Teaching Experience

Teaching Assistant at Paul G. Allen School of CSE, University of Washington
Natural Language Processing — Winter 2019

Teaching Assistant at EECS, Peking University

Introduction to Computer System	Fall 2015
Introduction to Computer System	Fall 2014

Professional Service

Journal reviewing: TACL (2021)

Program comittee member/reviewer: ACL Rolling Review (2021) ACL (2016–2021), EMNLP (2016–21), NAACL (2018, 2019, 2021), CoNLL (2019–2021), EACL (2017), ICLR (2019–2022), NeurIPS (2019–2021), ICML (2019–2021), KDD (2016)

Publications

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. ABC: Attention with Bounded-memory Control. Preprint, 2021.

Alexis Ross[1], Tongshuang Wu[1], **Hao Peng**, Matthew E. Peters, and Matt Gardner. Tailor: Generating and Perturbing Text with Semantic Controls. Preprint, 2021. [1] = equal contribution.

Jungo Kasai, **Hao Peng**, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. Finetuning Pretrained Transformers into RNNs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2021.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random Feature Attention. In *Proceedings of the Conference of the International Conference on Learning Representations*, 2021. **Spotlight.**

Jungo Kasai, Nikolaos Pappas, **Hao Peng**, James Cross, and Noah A. Smith. Deep Encoder, Shallow Decoder: Reevaluating the Speed-Quality Tradeoff in Machine Translation. In *Proceedings of the Conference of the International Conference on Learning Representations*, 2021.

Dianqi Li, Yizhe Zhang, **Hao Peng**, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. Contextualized Perturbation for Textual Adversarial Attack. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

Zhaofeng Wu, **Hao Peng**, and Noah A. Smith. Infusing Finetuning with Semantic Dependencies. *Transactions of the Association for Computational Linguistics*, 2020.

Hao Peng, Roy Schwartz, Dianqi Li, and Noah A. Smith. A Mixture of $h-1$ Heads is Better than h Heads. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2020.

Hao Peng, Roy Schwartz, and Noah A. Smith. PaLM: A Hybrid Parser and Language Model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

Jesse Dodge, Roy Schwartz, **Hao Peng**, and Noah A. Smith. RNN Architecture Learning with Sparse Regularization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

Hao Peng, Ankur P. Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. Text Generation with Exemplar-based Adaptive Decoding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. Rational Recurrences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2018.

Hao Peng, Sam Thomson, and Noah A. Smith. Backpropagating through Structured Argmax using a SPIGOT. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2018. **Honorable Mention for Best Paper Award.**

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. Learning Joint Semantic Parsers from Disjoint Data. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

Chenhao Tan, **Hao Peng**, and Noah A. Smith. "You are no Jack Kennedy": On Media Selection of Highlights from Presidential Debates. In *Proceedings of the International World Wide Web Conference*, 2018.

Hao Peng, Sam Thomson, and Noah A. Smith. Deep Multitask Learning for Semantic Dependency Parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2017.

Hao Peng, Jing Liu, and Chin-Yew Lin. News Citation Recommendation with Implicit and Explicit Semantics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2016.

Miltiadis Allamanis, **Hao Peng**, and Charles Sutton. A Convolutional Attention Network for Extreme Summarization of Source Code. In *Proceedings of the International Conference on Machine Learning*, 2016.

Hao Peng[1], Lili Mou[1], Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. Discriminative Neural Sentence Modeling by Tree-based Convolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015. [1] = equal contribution.

Hao Peng[1], Lili Mou[1], Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. A Comparative Study on Regularization Strategies for Embedding-based Neural Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015. [1] = equal contribution.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, **Hao Peng**, and Zhi Jin. Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

Hao Peng, Lili Mou, Ge Li, Yuxuan Liu, Lu Zhang, and Zhi Jin. Building Program Vector Representations for Deep Learning. In *Proceedings of International Conference on Knowledge Science, Engineering and Management*, 2015.

Skills

Programming Languages: Python, C/C++, CUDA, LaTeX
Maths & Statistical Package: PyTorch, JAX, TensorFlow, DyNet

References

Noah A. Smith (Doctoral Advisor)	
Amazon Professor of Machine Learning	UW Paul G. Allen School of CSE
Senior Research Manager	Allen Institute for AI
nasmith@cs.washington.edu	
Luke Zettlemoyer	
Professor	UW Paul G. Allen School of CSE
Research Scientist	Facebook AI Research
lsz@cs.washington.edu	
Yejin Choi	
Brett Helsel Professor	UW Paul G. Allen School of CSE
Senior Research Manager	Allen Institute for AI
yejin@cs.washington.edu	
Chris Dyer	
Research Scientist	DeepMind
Assistant Professor	CMU Language Technologies Institute
cdyer@google.com	

The ability to understand natural language is a foundation of artificial intelligence (AI). My long-term goal is to build **machine learning** models to advance artificial agents' ability to not only **perceive human language**, but also **acquire knowledge, reason, and communicate through it**. While the advent of large-scale deep learning models offers a seemingly clear path forward, challenges come along. The increasing computational overhead heightens the barriers to entry to cutting-edge natural language processing (NLP) research; the sophisticated deep learning models often generalize poorly to real-world settings, and it has become increasingly difficult to explain their decisions. My research aims to design **efficient, robust, generalizable, and interpretable representation learners for NLP**. During my Ph.D. career, I have taken several steps towards my long-term goal:

- **Efficient methods for NLP (§1):** I have developed and empirically validated efficient algorithms and learning paradigms for state-of-the-art NLP models [1, 2, 3, 4, 5].
- **Learning with structural inductive biases (§2):** my research has proposed a joint learning framework for meaning representations, and devised an algorithm to train NLP pipelines end-to-end. This linguistic knowledge in turn helps learn representations that are robust, generalizable, and sample-efficient [6, 7, 8, 9, 10, 11, 12].
- **Formal analysis of deep learning (§3):** I have formally characterized the connections between several modern neural architectures and weighted finite-state automata. The insights inspire accurate, efficient, and interpretable neural models [13, 14].

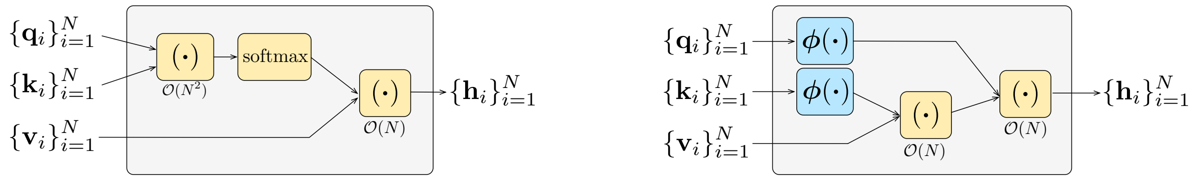
1 Efficient Methods for NLP

Artificial intelligence applications have seen unprecedented progress, which can be primarily attributed to the development of computationally-intensive deep learning models. However, their growing computational requirements have heightened the barriers to entry to state-of-the-art research and negatively impacted the environment. For example, searching for optimal architectures for certain NLP tasks [15] is estimated to cost more than 1 million USD, hardly affordable to less-resourced research groups. It emits a comparable amount of carbon dioxide as an average car does in 5 years [16]. My research aims to **improve the efficiency of state-of-the-art NLP models**, thereby **promoting the accessibility of cutting-edge NLP research, especially for less-funded institutions, and mitigate its environmental concerns.**

Attention with linear complexity. As one step towards this goal, I develop algorithms to improve the efficiency of the transformer architecture [17]. It is a crucial ingredient of recent advance in NLP and the backbone of many foundation models [18] such as BERT [19] and GPT-3 [20]. Transformers use the softmax attention to contextualize the input. Attention comes with a quadratic complexity in the input length and accounts for most of the overhead, especially for long sequences. Therefore, devising accurate and more efficient alternative contextualizations is the key to improving transformers' efficiency, which has become a focal point of the community [21]. I explore two directions to achieve this.

We derive a holistic view of several recent efficient transformer models [2]. Although seemingly disparate, they can be subsumed into one unified abstraction, "compressing" the context with various strategies, primarily using hand-crafted heuristics. This perspective not only connects several efficient attention variants that would otherwise appear apart but also gives fresh insights into established approaches, broadening their impact. Besides, it inspires a new algorithm that learns to compress the context in a context-dependent manner. On a variety of real-world NLP tasks, our model outperforms other efficient transformers.

Drawing inspiration from the classical kernel methods, I devise linear-complexity alternatives to softmax attention [1, 4]. Canonical attention calculates pairwise similarities between input tokens using a softmax normalization. RFA [1] devises a linear-complexity approximation to it using random feature techniques and a kernel trick (Figure 1). RFA significantly



(a) Softmax attention. (b) Random feature attention.

Figure 1: Computation graphs for softmax attention (left) and random feature attention (right), over inputs of length N. $\mathbf{q}_i$, $\mathbf{k}_i$, and $\mathbf{v}_i$ denote query, key, and value vectors, and $\mathbf{h}_i$ the outputs.

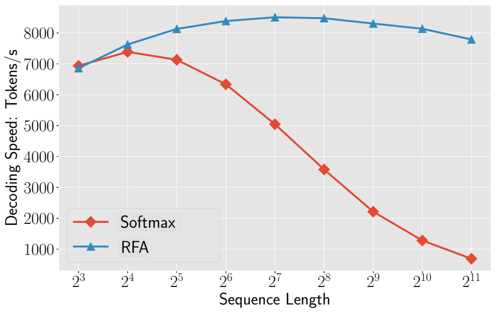
improves time and memory efficiency when applied in transformers, *without* any accuracy loss (e.g., 12× decoding speedup with less than 10% of the memory for 2,048-length sequences; Figure 2). Besides, it connects the attention to recurrent neural networks (RNNs), another important model family in NLP. Such connections allow borrowing existing wisdom from RNNs to enhance attention, which proves especially useful in document-level machine translation [5] and some applications in computer vision [22]. The empirical success of RFA opens new possibilities: a broader class of kernelized attention offers efficient alternatives to softmax attention. In a follow-up project [4], we observe that *learning* the kernel functions from data improves the accuracy-efficiency tradeoff. RFA was featured as a spotlight at ICLR 2021 and receives continuing interest from other research groups [23, 24, 22, 25].

Finetuning with efficient attention. Due to their sizes and expensive computation, large pretrained transformers are typically sub-optimal in settings with long inputs or limited computational resources. We propose a swap-then-finetune procedure [2, 4]: it replaces the softmax attention in pretrained transformers with its efficient counterparts (e.g., RFA) and then finetunes. In practice, this is an appealing approach since it avoids reinvesting the vast amounts of resources already put into pretraining. In an ongoing project, we explore training an efficient attention student from a transformer teacher using knowledge distillation.

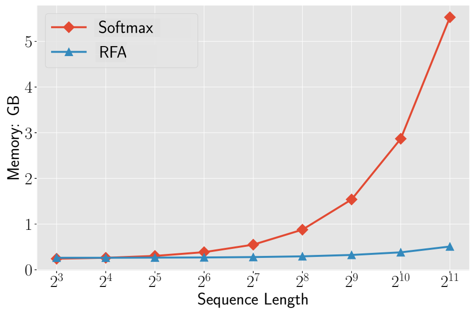
Future directions. A long-term goal of my research is to develop efficient methods for NLP. It helps promote the inclusiveness of cutting-edge NLP research and mitigate its negative environmental impact. From an algorithmic perspective, it requires meticulously diving into the mathematical details, rigorous implementations, and familiarity with modern hardware, which I have achieved in the past and will be excited to keep working on. Besides, I believe that getting insights from the tasks and datasets is crucial. For example, we found that simply pairing a shallow decoder with a deep encoder yields significantly faster machine translation decoding *without* accuracy loss [3]. Efficient machine learning has received increasing interest, and it is vital to draw broad and systematic connections among different lines of research to allow for the exchange of progress. I have done so in my previous works [2, 13] and am excited to expand such efforts.

2 Learning with Structural Inductive Biases

Language is structured: meanings are articulated and perceived through hierarchical compositions of words. However, NLP has witnessed a shift away from linguistic structures. In the past, they played a central role. For example, through an NLP pipeline, a sentence is first tagged with parts of speech, then parsed into a syntactic tree, then semantically analyzed, before being fed into a question answering system. Today, general-purpose representations pretrained on massive sequential data are finetuned on task-specific datasets. Although this new paradigm has brought unmatched empirical gains, its sample efficiency and robustness can be enhanced by explicitly encoding structures [9]. **Linguistic structures provide valuable inductive biases** that help models generalize to unseen scenarios; why has the community's



(a) Speed vs. lengths.



(b) Memory vs. lengths.

Figure 2: Decoding speed (left) and memory overhead (right) varying the output lengths.

interest in them been fading? Two primary reasons are: (1) for a long time, structured predictors were not accurate enough to produce useful features; (2) traditional NLP pipelines are less flexible than end-to-end training and prone to cascading errors. Answering to these challenges, my research has built **more accurate structured predictors for meaning representations** [6, 7], and proposed **algorithms to train NLP pipelines end-to-end** [8].

Jointly learning multiple meaning representations. Various theories of natural language semantics have been developed, reflecting different linguistic phenomena. They heavily rely on structures to represent meanings. The annotation of semantic structures, already expensive, is fragmented across competing theories. Although these representations may be structurally incompatible, they can be similar in spirit. For example (Figure 3), the phrase "*Only a few books*" fills a semantic role of the frame triggered by "*fell*" (bottom), and at the same time forms a subgraph semantically descending from it (top). We hypothesize that the overlap among different theories can be exploited using multitask learning, allowing us to learn from complementary resources jointly.

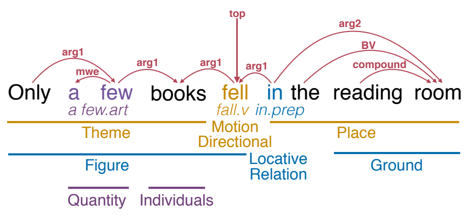


Figure 3: Structural similarities between two meaning representations: bilexical dependencies (top) and phrase-based frame semantics.

Our ACL 2017 work [6] utilizes the invaluable parallel annotations of three semantic formalisms [26]. We propose a joint decoding algorithm on top of other established multitask learning approaches (e.g., parameter sharing). Our results confirm the benefits of symbiosis among multiple semantic representations. This effort is extended to learning from disjoint (i.e., *no* parallel annotations) structurally more divergent resources [7]. These two works have received continuing interest from the community and are broadened by the Cross-Framework Meaning Representation Parsing shared tasks at CoNLL 2019 and 2020 [27, 28].[1] More broadly, they preceded a major shift towards multitask and transfer learning in NLP.

Learning NLP pipelines end-to-end. NLP pipelines run staged structured prediction modules to process text. They are prone to cascading errors and less amenable to neural learning—pipelines make discrete decisions at each stage, incompatible with end-to-end training using backpropagation. Our ACL 2018 work proposes SPIGOT [8], a stable and efficient algorithm to approximate the gradients for discrete structured argmax. SPIGOT calculates a proxy for the gradients **respecting the structured constraints**. As in Figure 4, SPIGOT introduces a projection step to keep the "updated" structures within the relaxed feasible set. SPIGOT allows backpropagating through structured prediction and using it as intermediate layers in neural networks, facilitating training NLP pipelines end-to-end. Our empirical results verify that using SPIGOT to incorporate "learnable" structural inductive biases yields more accurate and

[1] http://mrp.nlpl.eu/2020/index.php

interpretable decisions in downstream tasks. It can also be used in semi-supervised learning and unsupervised structure induction [29]. SPIGOT received a nomination for the best paper award at ACL 2018.

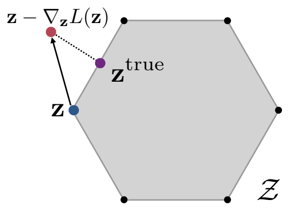

$\mathbf{z} - \nabla_{\mathbf{z}} L(\mathbf{z})$
$\mathbf{z}^{\text{true}}$
$\mathbf{z}$
$\mathcal{Z}$

Figure 4: Illustration of SPIGOT. The discrete feasible set is relaxed into a convex polytope $\mathcal{Z}$. $\nabla_{\mathbf{z}} L$, the gradients of loss with respect to predicted structures, are computed using backpropagation. If updating $\mathbf{z}$ makes it outside the polytope, it is projected back to $\mathcal{Z}$, resulting in $\mathbf{z}^{\text{true}}$.

Future directions. I am passionate about designing NLP models imbued with structural inductive biases. An important aspect is to build neural architectures reflecting linguistic structures, which I have done in my past research [10, 11, 30]. Looking forward, I want to focus on how linguistic structures interact with state-of-the-art attentive models like transformers. In an ongoing project, I am exploring training transformers with a structural prior over the attention distributions. Additionally, improving the efficiency and scalability of structured models is crucial to building more robust, generalizable, and interpretable NLP models in real-world settings. Recent advances in prompt learning provide new opportunities. For example, we bake semantic structural information into a pre-trained language model through prompting [12], *without* the efficiency challenges of explicitly encoding structures.

3 Formal Analysis of Deep Learning Models

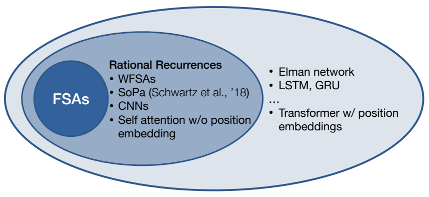

FSAs
Rational Recurrences
• WFSAs
• SoPa (Schwartz et al., '18)
• CNNs
• Self attention w/o position embedding
• Elman network
• LSTM, GRU
...
• Transformer w/ position embeddings

Figure 5: A formal hierarchy of established NLP models in terms of modeling capacities, induced by rational recurrences [31, 13, 32].

Despite their tremendous success in NLP, our understanding of deep learning models lags behind. An important part of my research **grounds modern neural architectures to formal principles in NLP** [13], and uses the insights to develop **better-performing, more efficient, and more interpretable models** [14, 33, 34].

Our EMNLP 2018 work [13] formally studies the connections between modern recurrent neural architectures and weighted finite-state automata (WFSAs). WFSAs naturally relate to pattern-matching methods, offering an intuitive way to explain their decisions (Figure 6); together with their siblings hidden Markov models, they are fundamental tools for NLP. Besides, WFSAs are well-studied, interpretable, efficient, and flexible to design. Our theoretical findings prove that a family of modern recurrent and convolutional neural networks (CNNs), at least in their single-layer cases, are WFSAs parameterized with neural networks, which we dub **rational recurrences**. The significance is both theoretical and empirical: rational recurrences provide a unifying framework for existing approaches and a new way to characterize their capacities (Figure 5); they lead to an algorithm that "translates" WFSAs into neural networks, offering a flexible way to devise new neural architectures imbued with desired inductive biases. Rational models are effective representation learners, as our empirical results show. In a follow-up work [14], we empirically show that rational neural models' decisions are interpretable (Figure 6), and that they are more parameter-efficient and well-suited for low-resource settings. For example, a rationally recurrent model trained with structured sparse regularization performs within 2% of the full model with fewer than 10% of the parameters.

Future directions. Looking ahead, I am passionate about improving our understanding of state-of-the-art deep learning models. Formal methods have been successfully applied to improving past generations of neural architectures; I am excited about extending this effort to attention-based models such as transformers. It will strengthen our understanding of attention, and how it connects to other established neural architectures such as RNNs and CNNs. Practically, it provides practitioners with a flexible way to design neural architectures imbued with desired inductive biases. As recent evidence suggests, a rationally recurrent model equipped with attention can be more accurate and efficient than state-of-the-art transformers [35]. I hope formal analysis and a shared framework for existing approaches could shed some light on future-generation neural networks, which, inevitably, will replace transformers.

Additionally, I aim to bridge the gap between formal principles and practice, and explore better learning paradigms based on the implications of linguistic theories. For example, we are working on a project to provide empirical evidence for the debate whether or not language models can learn meanings without direct supervision [36, 37]. Specifically, we craft grammars satisfying different levels of assumptions, and train language models on synthetic corpora sampled from those grammars and study to what extend they pick up meanings Preliminary results suggest interesting augmentations to the language modeling pretraining, which I am excited to explore.

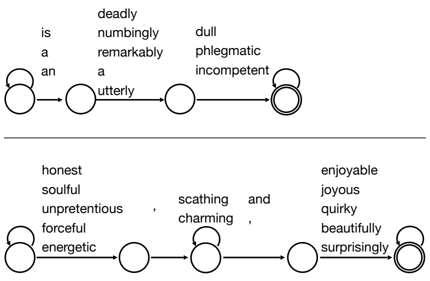


Figure 6: Patterns learned by a rational model, matching negative (top) and positive (bottom) sentiments.

4 Future Research Agenda

Large language models pretrained on massive data have shifted the paradigm of NLP. The empirical gains are genuinely remarkable. However, simply scaling up will hardly lead us to the ultimate goal of building AI with general linguistic capability. Perhaps a more important lesson is that the tasks, data, and evaluation that we thought to be difficult for machines are actually *not*. There has never been a better time for the community to start exploring machine learning models' capability to perform complex reasoning. I am excited to contribute to the reexamination of the tasks, annotation procedures, and evaluation protocols. Meanwhile, algorithmic progress is crucial. I believe that a better understanding of neural models through formal analysis can pave the way towards future neural architectures imbued with desired inductive biases, which are the key to building models that generalize better, are more robust and explainable, and are more sample-efficient to train. Equally importantly, devising efficient methods can help make cutting-edge research more accessible to less-funded institutions, promoting the openness and diversity of the AI community, the very reason it thrives.

Collaborations. Building next-generation NLP models requires joint forces across many fields. For example, my research on formal analysis of neural models complements theoretical approaches to explaining machine learning models' decisions. Learning more robust models relies on insights and inductive biases from linguistics and cognitive science. I am excited to collaborate with experts in these fields to build AI with better linguistic capabilities.

Efficient NLP is largely empowered by advances in hardware and machine learning systems. I plan to work with researchers in systems and architecture to devise algorithms that fully utilize the capacities of modern hardware, and build machine learning systems customized for NLP applications. I expect that the algorithmic aspects of my research expand beyond NLP. For example, I plan to expand my works on efficient transformers to modeling, e.g., DNA sequences or image data, which also require working with long sequences. More broadly, I look forward to working with experts in speech, computer vision, computational biology, reinforcement learning, and robotics to develop machine learning models that can better address the efficiency challenges in various applications.

References

[1] **Hao Peng**, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *Proc. of ICLR*, 2021.

[2] **Hao Peng**, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. ABC: Attention with bounded-memory control. *arXiv:2110.02488*, 2021. under review.

[3] Jungo Kasai, Nikolaos Pappas, **Hao Peng**, James Cross, and Noah Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *Proc. of ICLR*, 2021.

[4] Jungo Kasai, **Hao Peng**, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. Finetuning pretrained transformers into rnns. In *Proc. of EMNLP*, 2021.

[5] Zhaofeng Wu, **Hao Peng**, Nikolaos Pappas, and Noah A. Smith. Modeling context with linear attention for scalable document-level translation. 2021. Under review.

[6] **Hao Peng**, Sam Thomson, and Noah A. Smith. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*, 2017.

[7] **Hao Peng**, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*, 2018.

[8] **Hao Peng**, Sam Thomson, and Noah A. Smith. Backpropagating through structured argmax using a spigot. In *Proc. of ACL*, 2018.

[9] Zhaofeng Wu, **Hao Peng**, and Noah A. Smith. Infusing Finetuning with Semantic Dependencies. *TACL*, 9:226–242, 03 2021.

[10] Lili Mou, **Hao Peng**, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. Discriminative neural sentence modeling by tree-based convolution. In *Proc. of EMNLP*, 2015.

[11] Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, **Hao Peng**, and Zhi Jin. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. of EMNLP*, 2015.

[12] Alexis Ross, Tongshuang Wu, **Hao Peng**, Matthew E. Peters, and Matt Gardner. Tailor: Generating and perturbing text with semantic controls. *arXiv:2107.07150*, 2021. Under review.

[13] **Hao Peng**, Roy Schwartz, Sam Thomson, and Noah A. Smith. Rational recurrences. In *Proc. of EMNLP*, 2018.

[14] Jesse Dodge, Roy Schwartz, **Hao Peng**, , and Noah A. Smith. RNN architecture learning with sparse regularization. In *Proc. of EMNLP*, 2019.

[15] David R. So, Quoc V. Le, and Chen Liang. The evolved transformer. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proc. of ICML*, 2019.

[16] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proc. of ACL*, 2019.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NeurIPS*, 2017.

[18] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *arXiv:2108.07258*, 2021.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, 2019.

[20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, P. Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, Tom Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv:2005.14165*, 2020.

[21] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv: 2009.06732*, 2020.

[22] Lin Zheng, Huijie Pan, and Lingpeng Kong. Ripple attention for visual perception with sub-quadratic complexity. *arXiv:2110.02453*, 2021.

[23] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proc. of ICML*, 2021.

[24] Shengjie Luo, Shanda Li, Tianle Cai, Di He, Dinglan Peng, Shuxin Zheng, Guolin Ke, Liwei Wang, and Tie-Yan Liu. Stable, fast and accurate: Kernelized attention with relative positional encoding. In *Proc. of NeurIPS*, 2021.

[25] Sankalan Pal Chowdhury, Adamos Solomou, Avinava Dubey, and Mrinmaya Sachan. On learning the transformer kernel, 2021.

[26] Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*, 2015.

[27] Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. MRP 2019: Cross-framework meaning representation parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, 2019.

[28] Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, 2020.

[29] Tsvetomila Mihaylova, Vlad Niculae, and André F. T. Martins. Understanding the mechanics of SPIGOT: Surrogate gradients for latent structure learning. In *Proc. of EMNLP*, 2020.

[30] **Hao Peng**, Roy Schwartz, and Noah A. Smith. PaLM: A hybrid parser and language model. In *Proc. of EMNLP*, 2019.

[31] Roy Schwartz, Sam Thomson, and Noah A. Smith. SoPa: Bridging CNNs, RNNs, and weighted finite-state machines. In *Proc. of ACL*, 2018.

[32] William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. A formal hierarchy of RNN architectures. In *Proc. of ACL*, 2020.

[33] **Hao Peng**, Ankur P. Parikh, Manaal Faruqui, Bhuwan Dhingra, and Das Dipanjan. Text generation with exemplar-based adaptive decoding. In *Proc. of NAACL*, 2019.

[34] **Hao Peng**, Roy Schwartz, Dianqi Li, and Noah A. Smith. A mixture of h - 1 heads is better than h heads. In *Proc. of ACL*, 2020.

[35] Tao Lei. When attention meets fast recurrence: Training language models with reduced compute. In *Proc. of EMNLP*, 2021.

[36] Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proc. of ACL*, 2020.

[37] William Merrill, Yoav Goldberg, Roy Schwartz, and Noah A. Smith. Provable Limitations of Acquiring Meaning from Ungrounded Form: What Will Future Language Models Understand? *TACL*, 9:1047–1060, 2021.

The opportunity to engage in teaching and mentoring is my primary motivation to pursue an academic career. Teaching and mentoring are my responsibilities. I am grateful for the fascinating knowledge and guidance I received from my advisors and the community, and I aspire to pass them along and influence younger generations of researchers and engineers. Teaching is an invaluable opportunity for me to learn by having a more comprehensive understanding of the materials and from the talented and vigorous students. Building on my experience at UW, I am committed to improving my teaching and mentoring skills going forward.

Teaching and Lecturing

My teaching approach centers around several core principles. First, I emphasize big pictures and aim to provide holistic views of them when teaching new concepts. At UW, I assisted in a graduate-level natural language processing (NLP) course (with Prof. Noah Smith). One of the most challenging topics in the class was structured prediction. During my weekly office hour, I taught various algorithms with a unifying probabilistic model perspective. I also aim to highlight the historical context—through the years, NLP has made remarkable progress in developing new tools to parameterize the same set of probabilistic models; inference algorithms (e.g., the CKY algorithm) invented for statistical grammars decades ago are equally applicable in today's deep neural networks. Based on the feedback from the students, this perspective helped them learn the development process of a subject, and appreciate existing wisdom, and at the same time realize its limitations. The abstraction paves the way from classroom materials towards the students' research projects. An essential theme of my research is to study the connections among NLP models, which prepares me to teach different NLP topics through a systematic perspective in the future.

Second, I make explicit efforts to explore complementary lecture formats. In 2014 and 2015, I assisted an undergraduate-level course on computer systems at Peking University. Even for top CS-majoring students, it was a challenging class, covering both high-level concepts of modern computer architectures and low-level implementations in C and assembly language. During the weekly recitations, I complemented traditional lectures with quizzes, hands-on coding sessions, and flipped classrooms. The students found that the fresh formats kept them engaged and encouraged them to think about the course materials actively.

Third, I aim to balance between theory and practice—an AI course should train students to be both well-versed in theoretical concepts and proficient in implementation. As a teaching assistant for a graduate-level NLP course at UW, I designed the homework assignments. Through a set of problems, I asked the students to first go through several particular examples, then prove a more general proposition, and finally implement and train machine learning models on the data I assembled. I believe such hands-on components of the assignments help students quickly translate course materials into tools they can use in their research projects and real-world applications. Meanwhile, theory, abstractions, and fundamental principles teach students the necessary skills to self-educate in the future and adapt to the dramatically evolving AI technologies.

Invited lectures. Besides classroom instructions, I have given talks to audiences from different fields in both academia and industry, including seven invited talks and seven conference talks.[1] These experiences prepare me for giving course lectures targeting students from various fields and backgrounds.

[1]Invited talks at Peking University, DeepMind, Hongkong University 2021; the University of Alberta, DeepMind, 2020; Peking University 2018; NYU Shanghai 2017. Conference talks at ICLR 2021; ACL 2020; NAACL 2019; ACL, EMNLP, NAACL 2018; ACL 2016.

Going forward. I look forward to teaching both undergraduate and graduate courses on natural language processing, machine learning, deep learning, artificial intelligence, and other related areas. I am excited about developing new courses on emerging topics such as efficient machine learning and interpretability. My teaching experience makes me comfortable teaching courses of different forms, including lectures with a large audience and project or laboratory-focused courses with more hands-on programming practice. I am also excited to help organize research seminars and reading groups on topics related to my research.

Mentoring

During my graduate career at UW, I have had the privilege to mentor six talented undergraduate, Master's, and junior Ph.D. students from diverse backgrounds. *All* of the projects are led by the students, and many of them turned into papers published at or submitted to top-tier venues [1, 2, 3, 4]. As a mentor, I start by helping the students flesh out high-level ideas and research questions into concrete experiments. I break down long-term goals into small milestones, which we accomplish and discuss weekly. When it is helpful, I also help the students with mathematical details, implementations, and experiments. Finally, I help them in technical writing and conference presentations.

I try to establish long-term collaborations with my mentees, which prepares me to advise Ph.D. students as a professor. For example, I have been working with a UW Master's student for three years over four different projects. He was not experienced in NLP research when we started. In our first project on incorporating linguistic knowledge into representation learning, I gave him hands-on advice and helped with implementations. After publishing our findings at TACL [1], we had a conversation on the next steps. Knowing that he plans to pursue a Ph.D. in NLP, I encouraged him to leave the "comfort zone" and explore less-familiar topics. I believe that it would help him build a diverse skill set and find long-term research interests. Our second project is about efficient text generation [4]. I was supportive when he told me about his lack of confidence in his mathematical skills. We carefully went through the maths and technical details together. As a mentor, the most rewarding experience for me is to see that he grows more independent and better organized, and is now well-versed in many NLP topics. Our collaboration continues after he joined the Allen Institute for AI as a resident. He proposes new ideas and confidently drives the projects.

An important lesson I learned from my mentoring experience is that every student is unique. They have different strengths and learn in various ways. It is my role to cater my style according to theirs. I look forward to building open, collaborative mentoring relationships with my students, through which we can learn from each other.

References

[1] Zhaofeng Wu, **Hao Peng**, and Noah A. Smith. Infusing Finetuning with Semantic Dependencies. *TACL*, 9:226–242, 03 2021.

[2] Jungo Kasai, Nikolaos Pappas, **Hao Peng**, James Cross, and Noah Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *Proc. of ICLR*, 2021.

[3] Jungo Kasai, **Hao Peng**, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. Finetuning pretrained transformers into rnns. In *Proc. of EMNLP*, 2021.

[4] Zhaofeng Wu, **Hao Peng**, Nikolaos Pappas, and Noah A. Smith. Modeling context with linear attention for scalable document-level translation. 2021. Under review.

I deeply believe that a diverse and inclusive climate is vital, even more so in academia. It ensures equity, breeds creativity and innovation, and paves the path towards continuous excellence. In academia, implicit biases and unconscious exclusion can be easily concealed by various forms of barriers to entry, e.g., academic training, research impact, institutions' reputation. Open, transparent, and constructive discussions on diversity, equity, and inclusion (DEI) are beneficial; continued commitments and endeavors by the entire community are crucial.

I am proud to be part of Paul G. Allen School's efforts towards establishing a community of DEI.[1] I systematically learned how to prevent discrimination and build an inclusive environment through the Empowering Prevention and Inclusive Communities workshop at UW;[2] I actively participate in academic events aiming to promote the voices of underrepresented groups, such as the Widening NLP workshop at ACL 2020.[3] These experiences provided me with a concrete framework to combat prejudice, discrimination, and exclusion in practice.

During my Ph.D. career, I have had the invaluable chance to work with wonderful people from diverse ethnic groups and identities. I mentored students at various levels from different backgrounds. In my teaching, I aim to create a belonging and supportive environment. This commitment is reinforced by the presence of a hearing-impaired student in my class. She came to the United States for the education that she would *never* have the chance to receive in her home country due to a less welcoming environment towards disabled people.

As artificial intelligence (AI) and natural language processing (NLP) become increasingly intertwined with daily life, researchers need to be aware of and responsible for the societal impact of their works. Part of my research focuses on making NLP models socially-aware, and guarding against the potential social harm by AI systems. For example, we are developing an algorithm to "unlearn" gender and racial biases preexisting in the data, and inevitably picked up by pretrained language models such as BERT [1] and GPT [2]. I aspire to continuously improve AI technologies and make them more inclusive and accessible to people from all cultures and ethnic groups.

I believe that building an inclusive climate takes structural efforts that go beyond the campus. I received my pre-graduate education in a system where gender stereotypes are as prevalent as in the US, if not more. Many of the students' parents believe, *wrongly*, that "females are biologically less capable of studying science and technology, just as males are worse at arts and humanities." This cannot be further from the truth [3, 4, 5]. In 2017, I visited my high school and had the opportunity to talk to some students and their parents about the harm of gender biases and the roles they can play to combat the deeply-rooted stereotypes. I am committed to promoting diversity in the broader education system. As a professor, I plan to organize and participate in K-12 outreach programs to broaden the participation in computer science and STEM fields among minorities.

Looking ahead, I will carry on my mission of fostering a climate of diversity, equity, and inclusion. In the classroom, I plan to follow progress on pedagogical techniques to avoid unconscious biases. When designing the curricula, I would like to emphasize the academic contribution of researchers from historically underrepresented groups. As an advisor, I will make explicit efforts to recruit and retain talents from all cultural backgrounds and diverse identities. I will strive to create a supportive environment for my students and support broad collaborations among students from various backgrounds. I will also encourage them to participate in events and training on diversity, which I found immensely helpful. Being a professor at a renowned institution is a privilege. I pledge to use my resources to contribute to the collective endeavor towards a diverse and inclusive community.

[1] https://www.cs.washington.edu/diversity
[2] https://www.washington.edu/safecampus/epic-program/
[3] https://www.winlp.org

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, 2019.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, P. Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, Tom Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv:2005.14165*, 2020.

[3] Steven J. Spencer, Claude M. Steele, and Diane M. Quinn. Stereotype threat and women's math performance. *Journal of Experimental Social Psychology*, pages 4–28, 1999.

[4] Luigi Guiso, Ferdinando Monte, Paola Sapienza, and Luigi Zingales. Culture, gender, and math. *Science*, 320(5880):1164–1165, 2008.

[5] Hannah Fairfield and Alan McLean. Girls lead in science exam, but not in the United States. https://archive.nytimes.com/www.nytimes.com/interactive/2013/02/04/science/girls-lead-in-science-exam-but-not-in-the-united-states.html?hp, 2013. Accessed: 2021-11-10.

RANDOM FEATURE ATTENTION

Hao Peng[♠∗] **Nikolaos Pappas**[♠] **Dani Yogatama**[♣] **Roy Schwartz**[♡]
Noah A. Smith[♠◇] **Lingpeng Kong**[♦∗]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣]DeepMind [◇]Allen Institute for Artificial Intelligence
[♡]School of Computer Science & Engineering, Hebrew University of Jerusalem
[♦]Department of Computer Science , The University of Hong Kong
{hapeng,npappas,nasmith}@cs.washington.edu
dyogatama@google.com, roys@cs.huji.ac.il, lpk@cs.hku.hk

ABSTRACT

Transformers are state-of-the-art models for a variety of sequence modeling tasks. At their core is an attention function which models pairwise interactions between the inputs at every timestep. While attention is powerful, it does *not* scale efficiently to long sequences due to its quadratic time and space complexity in the sequence length. We propose RFA, a linear time and space **a**ttention that uses **r**andom **f**eature methods to approximate the softmax function, and explore its application in transformers. RFA can be used as a drop-in replacement for conventional softmax attention and offers a straightforward way of learning with recency bias through an optional gating mechanism. Experiments on language modeling and machine translation demonstrate that RFA achieves similar or better performance compared to strong transformer baselines. In the machine translation experiment, RFA decodes twice as fast as a vanilla transformer. Compared to existing efficient transformer variants, RFA is competitive in terms of both accuracy and efficiency on three long text classification datasets. Our analysis shows that RFA's efficiency gains are especially notable on long sequences, suggesting that RFA will be particularly useful in tasks that require working with large inputs, fast decoding speed, or low memory footprints.

1 INTRODUCTION

Transformer architectures (Vaswani et al., 2017) have achieved tremendous success on a variety of sequence modeling tasks (Ott et al., 2018; Radford et al., 2018; Parmar et al., 2018; Devlin et al., 2019; Parisotto et al., 2020, *inter alia*). Under the hood, the key component is attention (Bahdanau et al., 2015), which models pairwise interactions of the inputs, regardless of their distances from each other. This comes with quadratic time and memory costs, making the transformers computationally expensive, especially for long sequences. A large body of research has been devoted to improving their time and memory efficiency (Tay et al., 2020c). Although better *asymptotic* complexity and prominent gains for long sequences have been achieved (Lee et al., 2019; Child et al., 2019; Beltagy et al., 2020, *inter alia*), in practice, many existing approaches are less well-suited for moderate-length ones: the additional computation steps required by some approaches can overshadow the time and memory they save (Kitaev et al., 2020; Wang et al., 2020; Roy et al., 2020, *inter alia*).

This work proposes **r**andom **f**eature **a**ttention (RFA), an efficient attention variant that scales linearly in sequence length in terms of time and space, and achieves practical gains for both long and moderate length sequences. RFA builds on a kernel perspective of softmax (Rawat et al., 2019). Using the well-established random feature maps (Rahimi & Recht, 2007; Avron et al., 2016; §2), RFA approximates the dot-then-exponentiate function with a kernel trick (Hofmann et al., 2008): $\exp(\mathbf{x} \cdot \mathbf{y}) \approx \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y})$. Inspired by its connections to gated recurrent neural networks (Hochreiter & Schmidhuber, 1997; Cho et al., 2014) and fast weights (Schmidhuber, 1992), we further augment RFA with an optional gating mechanism, offering a straightforward way of learning with recency bias when locality is desired.

[∗]The majority of this work was done while these authors were at DeepMind.

RFA and its gated variant (§3) can be used as a drop-in substitute for the canonical softmax attention, and increase the number of parameters by less than 0.1%. We explore its applications in transformers on language modeling, machine translation, and long text classification (§4). Our experiments show that RFA achieves comparable performance to vanilla transformer baselines in all tasks, while outperforming a recent related approach (Katharopoulos et al., 2020). The gating mechanism proves particularly useful in language modeling: the gated variant of RFA outperforms the transformer baseline on WikiText-103. RFA shines in decoding, even for shorter sequences. In our head-to-head comparison on machine translation benchmarks, RFA decodes around $2\times$ faster than a transformer baseline, *without* accuracy loss. Comparisons to several recent efficient transformer variants on three long text classification datasets show that RFA is competitive in terms of both accuracy and efficiency. Our analysis (§5) shows that more significant time and memory efficiency improvements can be achieved for longer sequences: $12\times$ decoding speedup with less than 10% of the memory for 2,048-length outputs.

2 Background

2.1 Attention in Sequence Modeling

The attention mechanism (Bahdanau et al., 2015) has been widely used in many sequence modeling tasks. Its dot-product variant is the key building block for the state-of-the-art transformer architectures (Vaswani et al., 2017). Let $\{\mathbf{q}_t\}_{t=1}^N$ denote a sequence of N **query** vectors, that attend to sequences of M **key** and **value** vectors. At each timestep, the attention linearly combines the values weighted by the outputs of a softmax:

$$\operatorname{attn}\left(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}\right) = \sum_i \frac{\exp\left(\mathbf{q}_t \cdot \mathbf{k}_i / \tau\right)}{\sum_j \exp\left(\mathbf{q}_t \cdot \mathbf{k}_j / \tau\right)} \mathbf{v}_i^\top . \tag{1}$$

τ is the temperature hyperparameter determining how "flat" the softmax is (Hinton et al., 2015).[1]

Calculating attention for a single query takes $\mathcal{O}(M)$ time and space. For the full sequence of N queries the space amounts to $\mathcal{O}(MN)$. When the computation *cannot* be parallelized across the queries, e.g., in autoregressive decoding, the time complexity is quadratic in the sequence length.

2.2 Random Feature Methods

The theoretical backbone of this work is the unbiased estimation of the Gaussian kernel by Rahimi & Recht (2007). Based on Bochner's theorem (Bochner, 1955), Rahimi & Recht (2007) proposed random Fourier features to approximate a desired shift-invariant kernel. The method nonlinearly transforms a pair of vectors $\mathbf{x}$ and $\mathbf{y}$ using a **random feature map** $\boldsymbol{\phi}$; the inner product between $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{y})$ approximates the kernel evaluation on $\mathbf{x}$ and $\mathbf{y}$. More precisely:

Theorem 1 (Rahimi & Recht, 2007). *Let $\boldsymbol{\phi} : \mathbb{R}^d \rightarrow \mathbb{R}^{2D}$ be a nonlinear transformation:*

$$\boldsymbol{\phi}\left(\mathbf{x}\right) = \sqrt{1/D}\Big[\sin\left(\mathbf{w}_1 \cdot \mathbf{x}\right), \ldots, \sin\left(\mathbf{w}_D \cdot \mathbf{x}\right), \cos\left(\mathbf{w}_1 \cdot \mathbf{x}\right), \ldots, \cos\left(\mathbf{w}_D \cdot \mathbf{x}\right)\Big]^\top . \tag{2}$$

When d-dimensional random vectors $\mathbf{w}_i$ are independently sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$,

$$\mathbb{E}_{\mathbf{w}_i}\left[\boldsymbol{\phi}\left(\mathbf{x}\right) \cdot \boldsymbol{\phi}\left(\mathbf{y}\right)\right] = \exp\left(-\left\|\mathbf{x} - \mathbf{y}\right\|^2 / 2\sigma^2\right). \tag{3}$$

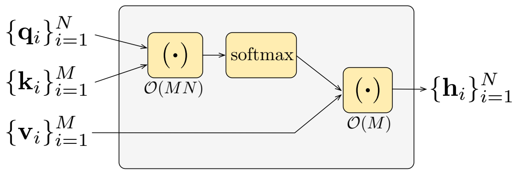
Variance of the estimation is inversely proportional to D (Appendix A.2; Yu et al., 2016).

Random feature methods proved successful in speeding up kernel methods (Oliva et al., 2015; Avron et al., 2017; Sun, 2019, *inter alia*), and more recently are used to efficiently approximate softmax (Rawat et al., 2019). In §3.1, we use it to derive an unbiased estimate to $\exp(\langle\cdot\,,\cdot\rangle)$ and then an efficient approximation to softmax attention.

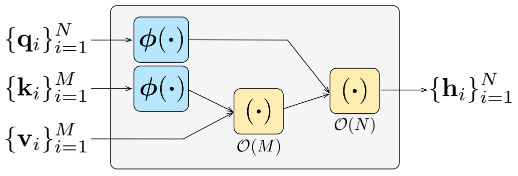
3 Model

This section presents RFA (§3.1) and its gated variant (§3.2). In §3.3 we lay out several design choices and relate RFA to prior works. We close by practically analyzing RFA's complexity (§3.4).

[1] $M = N$ in self-attention; they may differ, e.g., in the cross attention of a sequence-to-sequence model.


$\{\mathbf{q}_i\}_{i=1}^N$
$\{\mathbf{k}_i\}_{i=1}^M$
$\{\mathbf{v}_i\}_{i=1}^M$
$(\cdot)$
$\mathcal{O}(MN)$
softmax
$(\cdot)$
$\mathcal{O}(M)$
$\{\mathbf{h}_i\}_{i=1}^N$

(a) Softmax attention.


$\{\mathbf{q}_i\}_{i=1}^N$
$\{\mathbf{k}_i\}_{i=1}^M$
$\{\mathbf{v}_i\}_{i=1}^M$
$\phi(\cdot)$
$\phi(\cdot)$
$(\cdot)$
$\mathcal{O}(M)$
$(\cdot)$
$\mathcal{O}(N)$
$\{\mathbf{h}_i\}_{i=1}^N$

(b) Random feature attention.

Figure 1: Computation graphs for softmax attention (left) and random feature attention (right). Here, we assume cross attention with source length M and target length N.

3.1 RANDOM FEATURE ATTENTION

RFA builds on an unbiased estimate to $\exp(\langle \cdot\,, \cdot \rangle)$ from Theorem 1, which we begin with:

$$\begin{aligned}\exp\left(\mathbf{x}\cdot\mathbf{y}/\sigma^2\right) &= \exp\left(\|\mathbf{x}\|^2/2\sigma^2 + \|\mathbf{y}\|^2/2\sigma^2\right)\exp\left(-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2\right)\\ &\approx \exp\left(\|\mathbf{x}\|^2/2\sigma^2 + \|\mathbf{y}\|^2/2\sigma^2\right)\,\boldsymbol{\phi}(\mathbf{x})\cdot\boldsymbol{\phi}(\mathbf{y}).\end{aligned} \tag{4}$$

The last line does *not* have any nonlinear interaction between $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{y})$, allowing for a linear time/space approximation to attention. For clarity we assume the query and keys are unit vectors.[2]

$$\begin{aligned}\operatorname{attn}(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}) &= \sum_i \frac{\exp\left(\mathbf{q}_t\cdot\mathbf{k}_i/\sigma^2\right)}{\sum_j \exp\left(\mathbf{q}_t\cdot\mathbf{k}_j/\sigma^2\right)}\mathbf{v}_i^\top\\ &\approx \sum_i \frac{\boldsymbol{\phi}(\mathbf{q}_t)^\top\boldsymbol{\phi}(\mathbf{k}_i)\,\mathbf{v}_i^\top}{\sum_j \boldsymbol{\phi}(\mathbf{q}_t)\cdot\boldsymbol{\phi}(\mathbf{k}_j)}\\ &= \frac{\boldsymbol{\phi}(\mathbf{q}_t)^\top\sum_i\boldsymbol{\phi}(\mathbf{k}_i)\otimes\mathbf{v}_i}{\boldsymbol{\phi}(\mathbf{q}_t)\cdot\sum_j\boldsymbol{\phi}(\mathbf{k}_j)} = \textsc{Rfa}(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}).\end{aligned} \tag{5}$$

$\otimes$ denotes the outer product between vectors, and σ^2 corresponds to the temperature term τ in Eq. 1.

RFA can be used as a drop-in-replacement for softmax-attention.

(a) The input is revealed in full to **cross attention** and **encoder self-attention**. Here RFA calculates attention using Eq. 5.

(b) In **causal attention** RFA attends only to the prefix.[3] This allows for a recurrent computation. Tuple $(\mathbf{S}_t \in \mathbb{R}^{2D\times d}, \mathbf{z}_t \in \mathbb{R}^{2D})$ is used as the "hidden state" at time step t to keep track of the history, similar to those in RNNs. Then $\textsc{Rfa}(\mathbf{q}_t, \{\mathbf{k}_i\}_{i\leq t}, \{\mathbf{v}_i\}_{i\leq t}) = \boldsymbol{\phi}(\mathbf{q}_t)^\top\mathbf{S}_t/(\boldsymbol{\phi}(\mathbf{q}_t)\cdot\mathbf{z}_t)$, where

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{\phi}(\mathbf{k}_t)\otimes\mathbf{v}_t, \quad \mathbf{z}_t = \mathbf{z}_{t-1} + \boldsymbol{\phi}(\mathbf{k}_t). \tag{6}$$

$2D$ denotes the size of $\boldsymbol{\phi}(\cdot)$. Appendix A.1 summarizes the computation procedure of RFA, and Figure 1 compares it against the softmax attention. Appendix A.3 derives causal RFA in detail.

Analogously to the softmax attention, RFA has its multiheaded variant (Vaswani et al., 2017). In our experiments we use causal RFA in a transformer language model (§4.1), and both cross and causal RFA in the decoder of a sequence-to-sequence machine translation model.

3.2 RFA-GATE: LEARNING WITH RECENCY BIAS

The canonical softmax attention does *not* have any explicit modeling of distance or locality. In learning problems where such inductive bias is crucial (Ba et al., 2016; Parmar et al., 2018; Miconi et al., 2018; Li et al., 2019, *inter alia*), transformers heavily rely on positional encodings. Answering to this, many approaches have been proposed, e.g., learning the attention spans (Sukhbaatar et al.,

[2]This can be achieved by ℓ_2-normalizing the query and keys. See §3.3 for a related discussion.

[3]It is also sometimes called "decoder self-attention" or "autoregressive attention."

2019; Wu et al., 2020), and enhancing the attention computation with recurrent (Hao et al., 2019; Chen et al., 2019) or convolutional (Wu et al., 2019; Mohamed et al., 2019) components.

RFA faces the same issue, but its causal attention variant (Eq. 6) offers a straightforward way of learning with recency bias. We draw inspiration from its connections to RNNs, and augment RFA with a learned gating mechanism (Hochreiter & Schmidhuber, 1997; Cho et al., 2014; Peng et al., 2018, *inter alia*):

$$\begin{aligned} g_t &= \operatorname{sigmoid}(\mathbf{w}_g \cdot \mathbf{x}_t + b_g), \\ \mathbf{S}_t &= g_t\, \mathbf{S}_{t-1} + (1 - g_t)\, \boldsymbol{\phi}\left(\mathbf{k}_t\right) \otimes \mathbf{v}_t, \\ \mathbf{z}_t &= g_t\, \mathbf{z}_{t-1} + (1 - g_t)\, \boldsymbol{\phi}\left(\mathbf{k}_t\right). \end{aligned} \tag{7}$$

$\mathbf{w}_g$ and b_g are learned parameters, and $\mathbf{x}_t$ is the input representation at timestep t.[4] By multiplying the learned scalar gates $0 < g_t < 1$ against the hidden state $(\mathbf{S}_t, \mathbf{z}_t)$, history is exponentially decayed, favoring more recent context.

The gating mechanism shows another benefit of RFA: it would be otherwise more difficult to build similar techniques into the softmax attention, where there is no clear sense of "recurrence" (Appendix A.5). It proves useful in our language modeling experiments (§4.1).

3.3 DISCUSSION

On query and key norms, and learned random feature variance. Eq. 5 assumes both the query and keys are of norm-1. It therefore approximates a softmax attention that normalizes the queries and keys before multiplying them, and then scales the logits by dividing them by σ^2. Empirically, this normalization step scales down the logits (Vaswani et al., 2017) and enforces that $-1 \leq \mathbf{q}^\top \mathbf{k} \leq 1$. In consequence, the softmax outputs would be "flattened" if not for σ, which can be set *a priori* as a hyperparameter (Yu et al., 2016; Avron et al., 2017; Sun, 2019, *inter alia*). Here we instead learn it from data with the reparameterization trick (Kingma & Welling, 2014):

$$\widetilde{\mathbf{w}}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \quad \mathbf{w}_i = \boldsymbol{\sigma} \circ \widetilde{\mathbf{w}}_i. \tag{8}$$

$\mathbf{I}_d$ is the $d \times d$ identity matrix, and $\circ$ denotes elementwise product between vectors. d-dimensional vector $\boldsymbol{\sigma}$ is learned, but random vectors $\widetilde{\mathbf{w}}_i$ are *not*.[5]

This norm-1 constraint is never mandatory. Rather, we employ it for notation clarity and easier implementation. In preliminary experiments we find it has little impact on the performance when σ is set properly or learned from data. Eq. 12 in Appendix A presents RFA *without* imposing it.

Going beyond the Gaussian kernel. More broadly, random feature methods can be applied to a family of shift-invariant kernels, with the Gaussian kernel being one of them. In the same family, the order-1 arc-cosine kernel (Cho & Saul, 2009) can be approximated with feature map: $\boldsymbol{\phi}_{\text{arccos}}(\mathbf{x}) = \sqrt{1/D}[\operatorname{ReLU}(\mathbf{w}_1 \cdot \mathbf{x}), \ldots, \operatorname{ReLU}(\mathbf{w}_D \cdot \mathbf{x})]^\top$ (Alber et al., 2017).[6] In our experiments, the Gaussian and arc-cosine variants achieve similar performance. This supplements the exploration of alternatives to softmax in attention (Tsai et al., 2019; Gao et al., 2019).

Relations to prior work. Katharopoulos et al. (2020) inspire the causal attention variant of RFA. They use a feature map based on the exponential linear unit activation (Clevert et al., 2016): $\operatorname{elu}(\cdot) + 1$. It significantly *underperforms* both the baseline and RFA in our controlled experiments, showing the importance of a properly-chosen feature map. Random feature approximation of attention is also explored by a concurrent work (Choromanski et al., 2021), with applications in masked language modeling for proteins. They propose positive random features to approximate softmax, aiming for a lower variance in critical regions. RFA instead normalizes the queries and keys before random projection to reduce variance. Going beyond both, RFA establishes the benefits of random feature methods as a more universal substitute for softmax across all attention variants, facilitating its applications in, e.g., sequence-to-sequence learning.

[4]In multihead attention (Vaswani et al., 2017), $\mathbf{k}_t$ and $\mathbf{v}_t$ are calculated from $\mathbf{x}_t$ using learned affine transformations.

[5]This departs from Eq. 2 by lifting the isotropic assumption imposed on the Gaussian distribution: note the difference between the vector $\boldsymbol{\sigma}$ in Eq. 8 and the scalar σ in Eq. 3. We find this improves the performance in practice (§4), even though the same result in Theorem 1 may not directly apply.

[6]Apart from replacing the sinusoid functions with ReLU, it constructs $\mathbf{w}_i$ in the same way as Eq. 8.

There are interesting connections between gated RFA and fast weights (Schmidhuber, 1992; 1993; Ba et al., 2016; Miconi et al., 2018, *inter alia*). Emphasizing recent patterns, they learn a temporal memory to store history similarly to Eqs. 7. The main difference is that RFA additionally normalizes the output using $\boldsymbol{\phi}(\mathbf{q}_t) \cdot \mathbf{z}$ as in Eq. 6, a by-product of approximating softmax's partition function. It is intriguing to study the role of this normalization term, which we leave to future work.

3.4 Complexity Analysis

Time. Scaling linearly in the sequence lengths, RFA needs less computation (in terms of number of operations) for long sequences. This implies speedup wherever the quadratic-time softmax attention *cannot* be fully-parallelized across time steps. More specifically:

- Significant speedup can be expected in autoregressive *decoding*, both conditional (e.g., machine translation) and unconditional (e.g., sampling from a language model). For example, 1.9$\times$ speedup is achieved in our machine translation experiments (§4.2); and more for longer sequences (e.g., 12$\times$ for 2,048-length ones; §5).
- Some applications (e.g., language modeling, text classification) reveal inputs to the model in full.[7] When there are enough threads to parallelize softmax attention across time steps, hardly any speedup from RFA can be achieved; when there are not, typically for very long sequences (>1,000), substantial speed gain is possible. For example, RFA does *not* achieve any speedup when working with 512-length context (§4.1), but achieves a $5.3\times$ speedup with 4,000-length context (§4.2).

Memory. Asymptotically, RFA has a better memory efficiency than its softmax counterpart (linear vs. quadratic). To reach a more practical conclusion, we include in our analysis the cost of the feature maps. $\boldsymbol{\phi}$'s memory overhead largely depends on its size D. For example, let's consider the cross attention of a decoder. RFA uses $\mathcal{O}(4D + 2Dd)$ space to store $\boldsymbol{\phi}(\mathbf{q}_t)$, $\sum_i \boldsymbol{\phi}(\mathbf{k}_i) \otimes \mathbf{v}_i$, and $\sum_i \boldsymbol{\phi}(\mathbf{k}_i)$ (Eq. 5; line 12 of Algo. 2).[8] In contrast, softmax cross attention stores the encoder outputs with $\mathcal{O}(Md)$ memory, with M being the source length. In this case RFA has a lower memory overhead when $2D \ll M$. Typically D should be no less than d in order for reasonable approximation (Yu et al., 2016); In a transformer model, d is the size of an attention head, which is usually around 64 or 128 (Vaswani et al., 2017; Ott et al., 2018). This suggests that RFA can achieve significant memory saving with longer sequences, which is supported by our empirical analysis in §5. Further, using moderate sized feature maps is also desirable, so that its overhead does not overshadow the time and memory RFA saves. We experiment with D at d and $2d$; the benefit of using $D > 2d$ is marginal.

Appendix A.6 discusses the time and space complexity in more detail, and Appendix C.2 studies the effect of random feature size on performance.

4 Experiments

We evaluate RFA on language modeling, machine translation, and long text classification.

4.1 Language Modeling

Setting. We experiment with WikiText-103 (Merity et al., 2017). It is based on English Wikipedia. Table 5 in Appendix B summarizes some of its statistics. We compare the following models:

- BASE is our implementation of the strong transformer-based language model by Baevski & Auli (2019).
- RFA builds on BASE, but replaces the softmax attention with random feature attention. We experiment with both Gaussian and arc-cosine kernel variants.
- RFA-GATE additionally learns a sigmoid gate on top of RFA (§3.2). It also has a Gaussian kernel variant and a arc-cosine kernel one.[9]
- $\boldsymbol{\phi}_{\mathrm{elu}}$ is a baseline to RFA. Instead of the random feature methods it uses the $\mathrm{elu}(\cdot) + 1$ feature map, as in Katharopoulos et al. (2020).

[7] A causal masking is usually used to prevent the model from accessing future tokens in language models.

[8] RFA *never* constructs the $M \times 2D \times d$ tensor $[\boldsymbol{\phi}(\mathbf{k}_i) \otimes \mathbf{v}_i]_i$, but sequentially processes the sequence.

[9] This gating technique is specific to RFA variants, in the sense that it is less intuitive to apply it in BASE.

To ensure fair comparisons, we use comparable implementations, tuning, and training procedure. All models use a 512 block size during both training and evaluation, i.e., they read as input a segment of 512 consecutive tokens, *without* access to the context from previous mini-batches. RFA variants use 64-dimensional random feature maps. We experiment with two model size settings, **small** (around 38M parameters) and **big** (around 242M parameters); they are described in Appendix B.1 along with other implementation details.

	Small		Big	
Model	**Dev.**	**Test**	**Dev.**	**Test**
BASE	33.0	34.5	24.5	26.2
ϕ_{elu} (Katharopoulos et al., 2020)	38.4	40.1	28.7	30.2
RFA-Gaussian	33.6	35.7	25.8	27.5
RFA-arccos	36.0	37.7	26.4	28.1
RFA-GATE-Gaussian	**31.3**	**32.7**	**23.2**	**25.0**
RFA-GATE-arccos	**32.8**	**34.0**	24.8	26.3
RFA-GATE-Gaussian-Stateful	**29.4**	**30.5**	**22.0**	**23.5**

Table 1: Language model perplexity (lower is better) on the WikiText-103 development and test sets. Bolded numbers outperform BASE.

Results. Table 1 compares the models' performance in perplexity on WikiText-103 development and test data. Both kernel variants of RFA, *without* gating, outperform ϕ_{elu} by more than 2.4 and 2.1 test perplexity for the small and big model respectively, confirming the benefits from using random feature approximation.[10] Yet both *underperform* BASE, with RFA-Gaussian having a smaller gap. Comparing RFA against its gated variants, a more than 1.8 perplexity improvement can be attributed to the gating mechanism; and the gap is larger for small models. Notably, RFA-GATE-Gaussian outperforms BASE under both size settings by at least 1.2 perplexity. In general, RFA models with Gaussian feature maps outperform their arc-cosine counterparts.[11] From the analysis in §3.4 we would *not* expect speedup by RFA models, nor do we see any in the experiments.[12]

Closing this section, we explore a "stateful" variant of RFA-GATE-Gaussian. It passes the last hidden state $(\mathbf{S}_t, \mathbf{z}_t)$ to the next mini-batch during both training and evaluation, a technique commonly used in RNN language models (Merity et al., 2018). This is a consequence of RFA's RNN-style computation, and is less straightforward to be applicable in the vanilla transformer models.[13] From the last row of Table 1 we see that this brings a more than 1.5 test perplexity improvement.

4.2 MACHINE TRANSLATION

Datasets. We experiment with three standard machine translation datasets.

- WMT14 EN-DE and EN-FR (Bojar et al., 2014). Our data split and preprocessing follow those of Vaswani et al. (2017). We share the source and target vocabularies within each language pair, with 32,768 byte pair encoding types (BPE; Sennrich et al., 2016).
- IWSLT14 DE-EN (Cettolo et al., 2014) is based on TED talks. The preprocessing follows Edunov et al. (2018). Separate vocabularies of 9K/7K BPE types are used for the source and target.

Table 5 in Appendix B summarizes some statistics of the datasets.

[10]All models are trained for 150K steps; this could be part of the reason behind the suboptimal performance of ϕ_{elu}: it may need 3 times more gradient updates to reach similar performance to the softmax attention baseline (Katharopoulos et al., 2020).

[11]We observe that RFA Gaussian variants are more stable and easier to train than the arc-cosine ones as well as ϕ_{elu}. We conjecture that this is because the outputs of the Gaussian feature maps have an ℓ_2-norm of 1, which can help stabilize training. To see why, $\sin^2(x) + \cos^2(x) = \cos(x - x) = 1$.

[12]In fact, RFA *trains* around 15% slower than BASE due to the additional overhead from the feature maps.

[13]Some transformer models use a text segment from the previous mini-batch as a prefix (Baevski & Auli, 2019; Dai et al., 2019). Unlike RFA, this gives the model access to only a limited amount of context, and significantly increases the memory overhead.

Setting. We compare the RFA variants described in §4.1. They build on a BASE model that is our implementation of the base-sized transformer (Vaswani et al., 2017). All RFA models apply random feature attention in decoder cross and causal attention, but use softmax attention in encoders. This setting yields the greatest decoding time and memory savings (§3.4). We use 128/64 for D in cross/causal attention. RFA-GATE learns sigmoid gates in the decoder causal attention. The ϕ_{elu} baseline uses the same setting and applies feature map in both decoder cross and causal attention, but *not* in the encoders. Further details are described in Appendix B.2.

	WMT14		**IWSLT14**	
Model	**EN-DE**	**EN-FR**	**DE-EN**	**Speed**
BASE	28.1	39.0	34.6	$1.0\times$
ϕ_{elu} (Katharopoulos et al., 2020)	21.3	34.0	29.9	$2.0\times$
RFA-Gaussian	28.0	39.2	34.5	$1.8\times$
RFA-arccos	28.1	38.9	34.4	$1.9\times$
RFA-GATE-Gaussian	28.1	39.0	34.6	$1.8\times$
RFA-GATE-arccos	28.2	39.2	34.4	$1.9\times$

Table 2: Machine translation test set BLEU. The decoding speed (last column) is relative to BASE. All models are tested on a single TPU v2 accelerator, with batch size 32.

Results. Table 2 compares the models' test set BLEU on three machine translation datasets. Overall both Gaussian and arc-cosine variants of RFA achieve similar performance to BASE on all three datasets, significantly outperforming Katharopoulos et al. (2020). Differently from the trends in the language modeling experiments, here the gating mechanism does not lead to substantial gains. Notably, all RFA variants decode more than $1.8\times$ faster than BASE.

4.3 LONG TEXT CLASSIFICATION

We further evaluate RFA's accuracy and efficiency when used as text encoders on three NLP tasks from the recently proposed Long Range Arena benchmark (Tay et al., 2021), designed to evaluate efficient Transformer variants on tasks that require processing long sequences.[14]

Experimental setting and datasets. We compare RFA against baselines on the following datasets:

- ListOps (**LO**; Nangia & Bowman, 2018) aims to diagnose the capability of modelling hierarchically structured data. Given a sequence of operations on single-digit integers, the model predicts the solution, also a single-digit integer. It is formulated as a 10-way classification. We follow Tay et al. (2021) and consider sequences with 500–2,000 symbols.
- Character-level text classification with the **IMDb** movie review dataset (Maas et al., 2011). This is a binary sentiment classification task.
- Character-level document retrieval with the ACL Anthology Network (**AAN**; Radev et al., 2009) dataset. The model classifies whether there is a citation between a pair of papers.

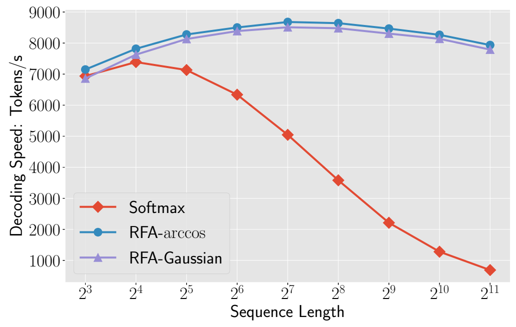
To ensure fair comparisons, we implement RFA on top of the transformer baseline by Tay et al. (2021), and closely follow their preprocessing, data split, model size, and training procedure. Speed and memory are evaluated on the IMDb dataset. For our RFA model, we use $D = 64$ for the IMDb dataset, and $D = 128$ for others. We refer the readers to Tay et al. (2021) for further details.

Results. From Table 3 we can see that RFA outperforms the transformer baseline on two out of the three datasets, achieving the best performance on IMDb with 66% accuracy. Averaging across three datasets, RFA outperforms the transformer by 0.3% accuracy, second only to Zaheer et al. (2020) with a 0.1% accuracy gap. In terms of time and memory efficiency, RFA is among the strongest. RFA speeds up over the transformer by 1.1–$5.3\times$, varying by sequence length. Importantly, compared to the only two baselines that perform comparably to the baseline transformer model (Tay et al., 2020a; Zaheer et al., 2020), RFA has a clear advantage in both speed and memory efficiency, and is the only model that is competitive in both accuracy and efficiency.

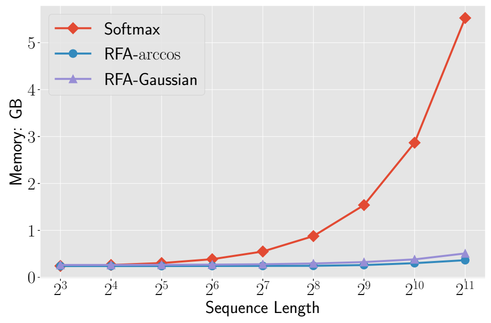
[14] `https://github.com/google-research/long-range-arena`

	Accuracy				Speed				Memory			
Model	**LO**	**IMDb**	**AAN**	**Avg.**	**1K**	**2K**	**3K**	**4K**	**1K**	**2K**	**3K**	**4K**
Transformer	36.4	64.3	57.5	52.7	1.0	1.0	1.0	1.0	1.00	1.00	1.00	1.00
Wang et al. (2020)	35.7	53.9	52.3	47.3	1.2	1.9	3.7	5.5	**0.44**	**0.21**	0.18	**0.10**
Kitaev et al. (2020)	**37.3**	56.1	53.4	48.9	0.5	0.4	0.7	0.8	0.56	0.37	0.28	0.24
Tay et al. (2020b)	17.1	63.6	**59.6**	46.8	1.1	1.6	2.9	3.8	0.55	0.31	0.20	0.16
Tay et al. (2020a)	37.0	61.7	54.7	51.1	1.1	1.2	2.9	1.4	0.76	0.75	0.74	0.74
Zaheer et al. (2020)	36.0	64.0	59.3	**53.1**	0.9	0.8	1.2	1.1	0.90	0.56	0.40	0.30
Katharopoulos et al. (2020)	16.1	65.9	53.1	45.0	1.1	**1.9**	3.7	5.6	0.44	0.22	**0.14**	0.11
Choromanski et al. (2021)	18.0	65.4	53.8	45.7	**1.2**	**1.9**	**3.8**	**5.7**	0.44	0.22	0.15	0.11
RFA-Gaussian (This work)	36.8	**66.0**	56.1	53.0	1.1	1.7	3.4	5.3	0.53	0.30	0.21	0.16

Table 3: Accuracy (higher is better) of different models on LO, IMDb, and AAN, along with their speed (higher is better) and peak memory consumption (lower is better) varying sequence lengths (1–4K). Speed and memory are evaluated on the IMDb dataset and relative to the transformer's. Bold font indicates the best performance in each column, and underlined numbers outperform the transformer in accuracy. Transformer's and previous works' numbers are due to Tay et al. (2021).



(a) Speed vs. lengths.



(b) Memory vs. lengths.

Figure 2: Conditional decoding speed (left) and memory overhead (right) varying the output lengths. All models are tested on a single TPU v2 accelerator, with greedy decoding and batch size 16.

5 Analysis

Decoding time and memory varying by sequence length. §3.4 shows that RFA can potentially achieve more significant speedup and memory saving for longer sequences, which we now explore.

We use a simulation conditional generation experiment on to compare RFA's sequence-to-sequence decoding speed and memory overhead against the baseline's. Here we assume the input and output sequences are of the same length. The compared models are of the same size as those described in §4.2, with 6-layer encoders and decoders. Other hyperparameters are summarized in Appendix B.2. All models are tested using greedy decoding with the same batch size of 16, on a TPU v2 accelerator.

From Figures 2 (a) and (b) we observe clear trends. Varying the lengths, both RFA variants achieve consistent decoding speed with nearly-constant memory overhead. In contrast, the baseline decodes slower for longer sequences, taking an increasing amount of memory. Notably, for 2,048-length sequences, RFA decodes around 12× faster than the baseline while using less than 10% of the memory. RFA-arccos slightly outperforms RFA-Gaussian in terms of speed and memory efficiency. This is because when using the same D (as we do here), the ϕ_{arccos} is half the size of ϕ_{Gaussian}. These results suggest that RFA can be particularly useful in sequence-to-sequence tasks with longer sequences, e.g., document-level machine translation (Miculicich et al., 2018).

Figure 3 in Appendix C.1 compares the speed and memory consumption in *unconditional* decoding (e.g., sampling from a language model). The overall trends are similar to those in Figure 2.

Notes on decoding speed. With a lower memory overhead, RFA can use a larger batch size than the baseline. As noted by Katharopoulos et al. (2020) and Kasai et al. (2021), if we had used mini-batches as large as the hardware allows, RFA could have achieved a more significant speed gain. Nonetheless, we control for batch size even though it is not the most favorable setting for RFA, since the conclusion translates better to common applications where one generates a single sequence at a time (e.g., instantaneous machine translation). For the softmax attention baseline, we follow Ott et al. (2018) and cache previously computed query/key/value representations, which significantly improves its decoding speed (over not caching).

Further analysis results. RFA achieves comparable performance to softmax attention. Appendix C.3 empirically shows that this *cannot* be attributed to RFA learning a good approximation to softmax: when we train with one attention but evaluate with the other, the performance is hardly better than randomly-initialized untrained models. Yet, an RFA model initialized from a pretrained softmax transformer achieves decent training loss after a moderate amount of finetuning steps (Appendix C.4). This suggests some potential applications, e.g., transferring knowledge from a pretrained transformer (e.g., GPT-3; Brown et al., 2020) to an RFA model that is more efficient to sample from.

6 RELATED WORK

One common motivation across the following studies, that is shared by this work and the research we have already discussed, is to scale transformers to long sequences. Note that there are plenty orthogonal choices for improving efficiency such as weight sharing (Dehghani et al., 2019), quantization (Shen et al., 2020), knowledge distillation (Sanh et al., 2020), and adapters (Houlsby et al., 2019). For a detailed overview we refer the reader to Tay et al. (2020c).

Sparse attention patterns. The idea behind these methods is to limit the reception field of attention computation. It motivates earlier attempts in improving attention's efficiency, and still receives lots of interest. The sparse patterns can be set *a priori* (Liu et al., 2018; Qiu et al., 2020; Ho et al., 2020; You et al., 2020, *inter alia*) or learned from data (Sukhbaatar et al., 2019; Roy et al., 2020, *inter alia*). For most of these approaches, it is yet to be empirically verified that they are suitable for large-scale sequence-to-sequence learning; few of them have recorded decoding speed benefits.

Compressed context. Wang et al. (2020) compress the context along the timesteps so that the effective sequence length for attention computation is reduced. Another line of work aims to store past context into a memory module with limited size (Lee et al., 2019; Ainslie et al., 2020; Rae et al., 2020, *inter alia*), so that accessing longer history only moderately increases the overhead. Reminiscent of RNN language models, RFA attends beyond a fixed context window through a stateful computation, *without* increasing time or memory overhead.

7 CONCLUSION

We presented random feature attention (RFA). It views the softmax attention through the lens of kernel methods, and approximates it with random feature methods. With an optional gating mechanism, RFA provides a straightforward way of learning with recency bias. RFA's time and space complexity is linear in the sequence length. We use RFA as a drop-in substitute for softmax attention in transformer models. On language modeling, machine translation, and long text classification benchmarks, RFA achieves comparable or better performance than strong baselines. In the machine translation experiment, RFA decodes twice as fast. Further time and memory efficiency improvements can be achieved for longer sequences.

ACKNOWLEDGMENTS

We would like to thank Phil Blunsom, Chris Dyer, Nando de Freitas, Jungo Kasai, Adhiguna Kuncoro, Dianqi Li, Ofir Press, Lianhui Qin, Swabha Swayamdipta, Sam Thomson, the language team at DeepMind and the ARK group at the University of Washington for their helpful feedback. We also thank Tay Yi for helping run the Long Range Arena experiments, Richard Tanburn for the advice on implementations, and the anonymous reviewers for their thoughtful comments. This work was supported in part by NSF grant 1562364 and a Google Fellowship. Nikolaos Pappas was supported by the Swiss National Science Foundation under grant number P400P2_183911 "UNISON."

REFERENCES

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In *Proc. of EMNLP*, 2020.

Maximilian Alber, Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Fei Sha. An empirical study on the properties of random bases for kernel methods. In *Proc. of NeurIPS*, 2017.

Haim Avron, Vikas Sindhwani, Jiyan Yang, and Michael W. Mahoney. Quasi-Monte Carlo feature maps for shift-invariant kernels. *Journal of Machine Learning Research*, 17(120):1–38, 2016.

Haim Avron, L. Kenneth Clarkson, and P. David and Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM J. Matrix Analysis Applications*, 2017.

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Proc. of NeurIPS*, 2016.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *Proc. of ICLR*, 2019.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*, 2015.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv: 2004.05150*, 2020.

S. Bochner. *Harmonic Analysis and the Theory of Probability*. University of California Press, 1955.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proc. of WMT*, 2014.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv: 2005.14165*, 2020.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign. In *Proc. of IWSLT*, 2014.

Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita. Recurrent positional embedding for neural machine translation. In *Proc. of EMNLP*, 2019.

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

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. In *Proc. of EMNLP*, 2014.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Proc. of NeurIPS*, 2009.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *Proc. of ICLR*, 2021.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proc. of ICLR*, 2016.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. of ACL*, 2019.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *Proc. of ICLR*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, 2019.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. Classical structured prediction losses for sequence to sequence learning. In *Proc. of NAACL*, 2018.

Yingbo Gao, Christian Herold, Weiyue Wang, and Hermann Ney. Exploring kernel functions in the softmax layer for contextual word classification. In *International Workshop on Spoken Language Translation*, 2019.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. Modeling recurrence for transformer. In *Proc. of NAACL*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NeurIPs Deep Learning and Representation Learning Workshop*, 2015.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv: 1912.12180*, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proc. of ICML*, 2019.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. In *Proc. of ICLR*, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proc. of ICML*, 2020.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. of ICLR*, 2014.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proc. of ICLR*, 2020.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proc. of ICML*, 2019.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Proc. of NeurIPS*, 2019.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *Proc. of ICLR*, 2018.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proc. of ACL*, 2011.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proc. of ICLR*, 2017.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and Optimizing LSTM Language Models. In *Proc. of ICLR*, 2018.

Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *Proc. of ICML*, 2018.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. Document-level neural machine translation with hierarchical attention networks. In *Proc. of EMNLP*, 2018.

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. Transformers with convolutional context for ASR. *arXiv: 1904.11660*, 2019.

Nikita Nangia and Samuel Bowman. ListOps: A diagnostic dataset for latent tree learning. In *Proc. of NAACL Student Research Workshop*, 2018.

Junier Oliva, William Neiswanger, Barnabas Poczos, Eric Xing, Hy Trac, Shirley Ho, and Jeff Schneider. Fast function to function regression. In *Proc. of AISTATS*, 2015.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proc. of WMT*, 2018.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. In *Proc. of ICML*, 2020.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *Proc. of ICML*, 2018.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. Rational recurrences. In *Proc. of EMNLP*, 2018.

Hao Peng, Roy Schwartz, Dianqi Li, and Noah A. Smith. A mixture of $h-1$ heads is better than h heads. In *Proc. of ACL*, 2020.

Matt Post. A call for clarity in reporting BLEU scores. In *Proc. of WMT*, 2018.

Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. In *Findings of EMNLP*, 2020.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. The ACL Anthology network. In *Proc. of the Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, 2009.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2018.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *Proc. of ICLR*, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proc. of NeurIPS*, 2007.

Ankit Singh Rawat, Jiecao Chen, Felix Xinnan X Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random Fourier features. In *Proc. of NeurIPS*, 2019.

Aurko Roy, Mohammad Taghi Saffar, David Grangier, and Ashish Vaswani. Efficient content-based sparse attention with routing transformers. *arXiv: 2003.05997*, 2020.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv: 1910.01108*, 2020.

J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

J. Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *Proc. of ICANN*, 1993.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proc. of ACL*, 2016.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Q-BERT: Hessian based ultra low precision quantization of BERT. In *Proc. of AAAI*, 2020.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proc. of ACL*, 2019.

Yitong Sun. *Random Features Methods in Supervised Learning*. PhD thesis, The University of Michigan, 2019.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *arXiv: 2005.00743*, 2020a.

Yi Tay, Dara Bahri, Liu Yang, Don Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *Proc. of ICML*, 2020b.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv: 2009.06732*, 2020c.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *Proc. of ICLR*, 2021.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proc. of EMNLP*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NeurIPS*, 2017.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv: 2006.04768*, 2020.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *Proc. of ICLR*, 2019.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. In *Proc. of ICLR*, 2020.

Weiqiu You, Simeng Sun, and Mohit Iyyer. Hard-coded Gaussian attention for neural machine translation. In *Proc. of ACL*, 2020.

Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *Proc. of NeurIPS*, 2016.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *arXiv: 2007.14062*, 2020.

Appendices

A Random Feature Attention in More Detail

A.1 Detailed Computation Procedure

Algorithms 1 and 2 describe causal and cross random feature attention's computation procedures.

Algorithm 1 Causal random feature attention.

```
 1: procedure RFA-CAUSAL( $\{\mathbf{q}_i\}_{i=1}^N$, $\{\mathbf{k}_i\}_{i=1}^N$, $\{\mathbf{v}_i\}_{i=1}^N$)
 2:     ▷ $\mathbf{S}$ is a $D \times d$ matrix
 3:     ▷ $\mathbf{z}$ is a $D$-dimensional vector
 4:     $\mathbf{S}, \mathbf{z} \leftarrow \mathbf{0}, \mathbf{0}$
 5:     for $i = 1$ to $N$ do
 6:         $\widetilde{\mathbf{q}}_i, \widetilde{\mathbf{k}}_i \leftarrow \boldsymbol{\phi}(\mathbf{q}_i), \boldsymbol{\phi}(\mathbf{k}_i)$    ▷ Random feature maps
 7:         $\mathbf{S} \leftarrow \mathbf{S} + \widetilde{\mathbf{k}}_i \otimes \mathbf{v}_i$
 8:         $\mathbf{z} \leftarrow \mathbf{z} + \widetilde{\mathbf{k}}_i$
 9:         $\mathbf{h}_i^\top \leftarrow \widetilde{\mathbf{q}}_i^\top \mathbf{S} / (\widetilde{\mathbf{q}}_i \cdot \mathbf{z})$
10:     end for
11:     return $\{\mathbf{h}_i\}_{i=1}^N$
12: end procedure
```

Algorithm 2 Cross random feature attention.

```
 1: procedure RFA-CROSS( $\{\mathbf{q}_i\}_{i=1}^N$, $\{\mathbf{k}_i\}_{i=1}^M$, $\{\mathbf{v}_i\}_{i=1}^M$)
 2:     ▷ $\mathbf{S}$ is a $D \times d$ matrix
 3:     ▷ $\mathbf{z}$ is a $D$-dimensional vector
 4:     $\mathbf{S}, \mathbf{z} \leftarrow \mathbf{0}, \mathbf{0}$
 5:     for $i = 1$ to $M$ do
 6:         $\widetilde{\mathbf{k}}_i \leftarrow \boldsymbol{\phi}(\mathbf{k}_i)$    ▷ Random feature map
 7:         $\mathbf{S} \leftarrow \mathbf{S} + \widetilde{\mathbf{k}}_i \otimes \mathbf{v}_i^\top$
 8:         $\mathbf{z} \leftarrow \mathbf{z} + \widetilde{\mathbf{k}}_i$
 9:     end for
10:     for $i = 1$ to $N$ do
11:         $\widetilde{\mathbf{q}}_i \leftarrow \boldsymbol{\phi}(\mathbf{q}_i)$    ▷ Random feature map
12:         $\mathbf{h}_i^\top \leftarrow \widetilde{\mathbf{q}}_i^\top \mathbf{S} / (\widetilde{\mathbf{q}}_i \cdot \mathbf{z})$
13:     end for
14:     return $\{\mathbf{h}_i\}_{i=1}^N$
15: end procedure
```

A.2 Variance of Random Fourier Features

The following result is due to Yu et al. (2016). Using the same notation as in §2.2:

$$\mathrm{Var}(\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y})) = \frac{1}{2D}\left(1 - e^{-z^2}\right)^2, \tag{9}$$

where $z = \|\mathbf{x} - \mathbf{y}\| / \sigma$.

A.3 Derivation of Causal Rfa

This section presents a detailed derivation of causal RFA as in §3.1. Following Eq. 5 but changing the attended keys and values to the prefix:

$$\text{RFA}(\mathbf{q}_t, \{\mathbf{k}_i\}_{i \le t}, \{\mathbf{v}_i\}_{i \le t}) = \frac{\boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \sum_{i \le t} \boldsymbol{\phi}\left(\mathbf{k}_i\right) \otimes \mathbf{v}_i}{\boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \sum_{j \le t} \boldsymbol{\phi}\left(\mathbf{k}_j\right)} \tag{10}$$

Let $\mathbf{S}_t \triangleq \sum_{i \le t} \boldsymbol{\phi}(\mathbf{k}_i) \otimes \mathbf{v}_i$, and $\mathbf{z}_t \triangleq \sum_{i \le t} \boldsymbol{\phi}(\mathbf{k}_i)$; both can be calculated recurrently. Assuming $\mathbf{S}_0 = \mathbf{0}$ and $\mathbf{z}_0 = \mathbf{0}$:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{\phi}\left(\mathbf{k}_t\right) \otimes \mathbf{v}_t, \quad \mathbf{z}_t = \mathbf{z}_{t-1} + \boldsymbol{\phi}\left(\mathbf{k}_t\right), \quad t \ge 1. \tag{11}$$

This completes the derivation of causal RFA as in §3.1.

A.4 Rfa without Norm-1 Constraints

§3.1 assumes that the queries and keys are unit vectors. This norm-1 constraint is *not* a must. Here we present a RFA *without* imposing this constraint. Let $C(\mathbf{x}) = \exp(\|\mathbf{x}\|^2 / 2\sigma^2)$. From Eq. 4 we have $\operatorname{attn}\left(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}\right) =$

$$\begin{aligned}\sum_i \frac{\exp\left(\mathbf{q}_t \cdot \mathbf{k}_i / \sigma^2\right)}{\sum_j \exp\left(\mathbf{q}_t \cdot \mathbf{k}_j / \sigma^2\right)} \mathbf{v}_i^\top &\approx \sum_i \frac{C(\mathbf{q}_t)\, C(\mathbf{k}_i)\, \boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \boldsymbol{\phi}\left(\mathbf{k}_i\right) \mathbf{v}_i^\top}{\sum_j C(\mathbf{q}_t)\, C(\mathbf{k}_j)\, \boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \boldsymbol{\phi}\left(\mathbf{k}_j\right)} \\ &= \frac{\boldsymbol{\phi}\left(\mathbf{q}_t\right)^\top \sum_i C(\mathbf{k}_i)\, \boldsymbol{\phi}\left(\mathbf{k}_i\right) \otimes \mathbf{v}_i}{\boldsymbol{\phi}\left(\mathbf{q}_t\right) \cdot \sum_j C(\mathbf{k}_j)\, \boldsymbol{\phi}\left(\mathbf{k}_j\right)}.\end{aligned} \tag{12}$$

The specific attention computation is similar to those in §3.1. In sum, lifting the norm-1 constraint brings an additional scalar term $C(\cdot)$.

A.5 Relating Rfa-Gate to Softmax Attention

Drawing inspiration from gated RNNs, §3.2 introduces a gated variant of RFA. Now we study its "softmax counterpart."

$$\begin{aligned}&\widetilde{\mathbf{k}}_i = \mathbf{k}_i\left(1 - g_i\right) \prod_{j=i+1}^{t} g_j, \quad \widetilde{\mathbf{v}}_i = \mathbf{v}_i\left(1 - g_i\right) \prod_{j=i+1}^{t} g_j, \quad i = 1, \dots, t \\ &\mathbf{h}_t = \operatorname{attn}(\mathbf{q}_t, \{\widetilde{\mathbf{k}}_i\}_{i \le t}, \{\widetilde{\mathbf{v}}_i\}_{i \le t}).\end{aligned} \tag{13}$$

$\mathbf{h}_t$ is the output at timestep t and is used for onward computation.

At each step, all prefix keys and values are decayed by a gate value before calculating the attention. This implies that the attention computation for $\mathbf{q}_{t+1}$ *cannot* start until that of $\mathbf{q}_t$ is finished. Combined with the linear complexity of softmax normalization, this amounts to quadratic time in sequence length, even for language modeling training.

The above model is less intuitive and more expensive in practice, without the RFA perspective. This shows that RFA brings some benefits in developing new attention models.

A.6 Detailed Complexity Analysis

Table 4 considers a sequence-to-sequence model, and breaks down the comparisons to training (with teacher forcing; Williams & Zipser, 1989) and autoregressive decoding. Here we assume enough threads to fully parallelize softmax attention across timesteps when the inputs are revealed to the model in full. RFA has a lower space complexity, since it never explicitly populates the attention matrices. As for time, RFA trains in linear time, and so does the softmax attention: in teacher-forcing training a standard transformer decoder parallelizes the attention computation across time steps. The trend of the time comparison differs during decoding: when only one output token is produced at a time, RFA decodes linearly in the output length, while softmax attention decodes quadratically.

		Time Complexity			Space Complexity		
Setting	**Model**	**Encoder**	**Cross**	**Causal**	**Encoder**	**Cross**	**Causal**
Training w/ teacher forcing	softmax	$\mathcal{O}(M)$	$\mathcal{O}(M)$	$\mathcal{O}(N)$	$\mathcal{O}(M^2)$	$\mathcal{O}(MN)$	$\mathcal{O}(N^2)$
	RFA	$\mathcal{O}(M)$	$\mathcal{O}(M)$	$\mathcal{O}(N)$	$\mathcal{O}(M)$	$\mathcal{O}(M+N)$	$\mathcal{O}(N)$
Decoding	softmax	$\mathcal{O}(M)$	$\mathcal{O}(MN)$	$\mathcal{O}(N^2)$	$\mathcal{O}(M^2)$	$\mathcal{O}(MN)$	$\mathcal{O}(N^2)$
	RFA	$\mathcal{O}(M)$	$\mathcal{O}(M+N)$	$\mathcal{O}(N)$	$\mathcal{O}(M)$	$\mathcal{O}(M+N)$	$\mathcal{O}(N)$

Table 4: Time and space complexity comparisons between RFA and its softmax counterpart in a sequence-to-sequence attentive model, assuming an infinite amount of available threads. M and N denote the lengths of the source and target sequences respectively. Teacher forcing training (Williams & Zipser, 1989) and autoregressive decoding are assumed. Blue color indicates the cases where RFA asymptotically outperforms softmax attention.

Data	**Train**	**Dev.**	**Test**	**Vocab.**
WikiText-103	103M	218K	246K	268K
WMT14 EN-DE	4.5M	3K	3K	32K
WMT14 EN-FR	4.5M	3K	3K	32K
IWSLT14 DE-EN	160K	7K	7K	9K/7K

Table 5: Some statistics for the datasets. WikiText-103 split sizes are in number of tokens, while others are in number of instances.

B Experimental Details

Table 5 summarizes some statistics of the datasets used in our experiments. Our implementation is based on JAX.[15]

# Random Matrices	1	50	100	200
BLEU	24.0	25.7	25.8	25.8

Table 6: WMT14 EN-DE development set performance varying the number of random matrices to sample from during training. No beam search or checkpoint averaging is used.

During training, we sample a different random projection matrix for each attention head. Preliminary experiments suggest this performs better than using the same random projection throughout training (Table 6). Our conjecture is that this helps keep the attention heads from "over committing" to any particular random projection (Peng et al., 2020). To avoid the overhead of sampling from Gaussian during training, we do this in an offline manner. I.e., before training we construct a pool of random matrices (typically 200), at each training step we draw from the pool. At test time each attention head uses the same random projection, since no accuracy benefit is observed by using different ones for different test instances.

B.1 Language Modeling

We compare the models using two model size settings, summarized in Table 7. We use the fixed sinusoidal position embeddings by Vaswani et al. (2017). All models are trained for up to 150K gradient steps using the Adam optimizer (Kingma & Ba, 2015). No ℓ_2-regularization is used. We apply early stopping based on development set perplexity. All models are trained using 16 TPU v3 accelerators, and tested using a single TPU v2 accelerator.

[15] `https://github.com/google/jax`.

Hyperprams.	**Small**	**Big**
# Layers	6	16
# Heads	8	16
Embedding Size	512	1024
Head Size	64	64
FFN Size	2048	4096
Batch Size	64	64
Learning Rate	$[1 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}]$	
Warmup Steps	6000	6000
Gradient Clipping Norm	0.25	0.25
Dropout	[0.05, 0.1]	[0.2, 0.25, 0.3]
Random Feature Map Size	64	64

Table 7: Hyperparameters used in the language modeling experiments.

B.2 Machine Translation

WMT14. We use the fixed sinusoidal position embeddings by Vaswani et al. (2017). For both EN-DE and EN-FR experiments, we train the models using the Adam (with $\beta_1 = 0.1$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$) optimizer for up to 350K gradient steps. We use a batch size of 1,024 instances for EN-DE, while 4,096 for the much larger EN-FR dataset. The learning rate follows that by Vaswani et al. (2017). Early stopping is applied based on development set BLEU. No ℓ_2 regularization or gradient clipping is used. All models are trained using 16 TPU v3 accelerators, and tested using a single TPU v2 accelerator. Following standard practice, we average 10 most recent checkpoints at test time. We evaluate the models using SacreBLEU (Post, 2018).[16] A beam search with beam size 4 and length penalty 0.6 is used. Other hyperparameters are summarized in Table 8.

Hyperprams.	**WMT14**	**IWSLT14**
# Layers	6	6
# Heads	8	8
Embedding Size	512	512
Head Size	64	64
FFN Size	2048	2048
Warmup Steps	6000	4000
Dropout	0.1	0.3
Cross Attention Feature Map	128	128
Causal Attention Feature Map	64	64

Table 8: Hyperparameters used in the machine translation experiments.

C More Analysis Results

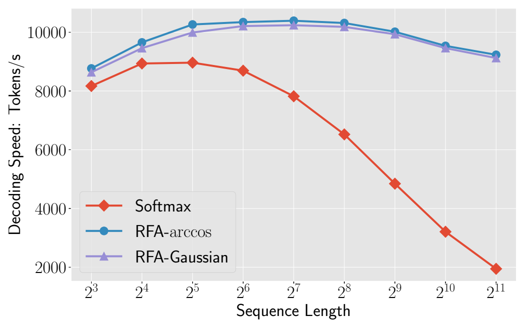
C.1 More Results on Decoding Speed and Memory Overhead

Figure 3 compares the RFA's *unconditional* decoding speed and memory against the softmax attention. The setting is the same as that in §5 except that here the models do not have an encoder. This experiment aims to simulate the applications such as sampling from a language model.

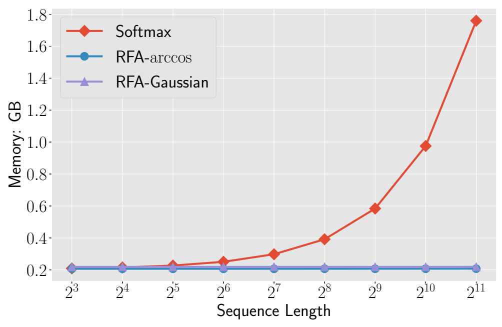
C.2 Effect of Random Feature Size

This section studies how the size of $\phi(\cdot)$ affects the performance. Table 9 summarize RFA-Gaussian's performance on WMT14 EN-DE development set. The model and training are the same as that used in §4.2 except random feature size. Recall from §2.2 that the size of $\phi(\cdot)$ is $2D$ for

[16] https://github.com/mjpost/sacrebleu



(a) Speed vs. lengths.



(b) Memory vs. lengths.

Figure 3: Unconditional decoding speed (left) and memory overhead (right) varying the output lengths. All models are tested on a single TPU v2 accelerator, with greedy decoding and batch size 16.

RFA-Gaussian. When the size of $\phi(\cdot)$ is too small (32 or 64 for cross attention, 32 for causal attention), training does not converge. We observe accuracy improvements by using random features sufficiently large (256 for cross attention and 128 for causal attention); going beyond that, the benefit is marginal.

ϕ **Size**	32	64	128	256	512
BLEU	N/A	N/A	24.9	25.8	26.0

(a) Varying cross attention ϕ sizes while fixing that of causal attention to be 128.

ϕ **Size**	32	64	128	256	512
BLEU	N/A	25.3	25.8	25.8	25.6

(b) Varying causal attention ϕ sizes while fixing that of cross attention to be 256.

Table 9: WMT14 EN-DE development set performance of RFA-Gaussian (the size of ϕ is $2D$; §2.2) varying the random feature sizes. N/A indicates training does not converge. No beam search or checkpoint averaging is used.

C.3 Train and Evaluate with Different Attention Functions

RFA achieves comparable performance to its softmax counterpart. Does this imply that it learns a good approximation to the softmax attention? To answer this question, we consider:

(i) an RFA-Gaussian model initialized from a pretrained softmax-transformer;
(ii) a softmax-transformer initialized from a pretrained an RFA-Gaussian model.

If RFA's good performance can be attributed to learning a good approximation to softmax, both, *without* finetunining, should perform similarly to the pretrained models. However, this is *not* the case on IWSLT14 DE-EN. Both pretrained models achieve more than 35.2 development set BLEU. In contrast, (i) and (ii) respectively get 2.3 and 1.1 BLEU *without* finetuning, hardly beating a randomly-initialized untrained model. This result aligns with the observation by Choromanski et al. (2021), and suggests that it is *not* the case that RFA performs well because it learns to imitate softmax attention's outputs.

C.4 Knowledge Transfer from Softmax Attention to RFA

We first supplement the observation in Appendix C.3 by finetuning (i) on the same pretraining data. Figure 4 plots the learning curves. It takes RFA roughly 1,500 steps to reach similar training loss to the pretrained model. As a baseline, "RFA Reset" resets the multihead attention parameters (i.e., those for query, key, value, and output projections) to randomly initialized ones. Its learning curve is similar to that of (i), suggesting that the pretrained multihead attention parameters are no more useful to RFA than randomly initialized ones. To further confirm this observation, "softmax Reset"

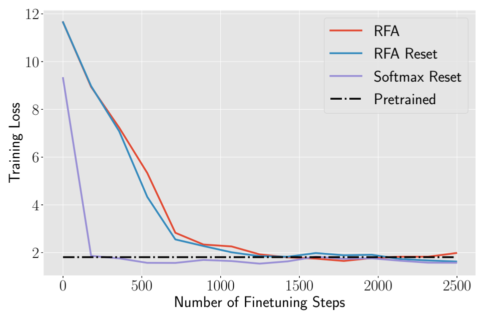


Figure 4: Finetuning an RFA-Gaussian model with its parameters initialized from a pretrained softmax-transformer. "Reset" indicates resetting the multihead attention parameters to randomly-initialized ones. The dashed line indicates the training loss of the pretrained model.

resets the multihead attention parameters *without* changing the attention functions. It converges to the pretraining loss in less than 200 steps.

Takeaway. From the above results on IWSLT14, pretrained knowledge in a softmax transformer *cannot* be directly transferred to an RFA model. However, from Figure 4 and a much larger-scale experiment by Choromanski et al. (2021), we do observe that RFA can recover the pretraining loss, and the computation cost of finetuning is much less than training a model from scratch. This suggests some potential applications. For example, one might be able to initialize an RFA language model from a softmax transformer pretrained on large-scale data (e.g., GPT-3; Brown et al., 2020), and finetune it at a low cost. The outcome would be an RFA model retaining most of the pretraining knowledge, but is much faster and more memory-friendly to sample from. We leave such exploration to future work.

Backpropagating through Structured Argmax using a SPIGOT

Hao Peng[◊] **Sam Thomson**[♣] **Noah A. Smith**[◊]
[◊] Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣] School of Computer Science, Carnegie Mellon University
{hapeng,nasmith}@cs.washington.edu, sthomson@cs.cmu.edu

Abstract

We introduce the **structured projection of intermediate gradients** optimization technique (SPIGOT), a new method for backpropagating through neural networks that include hard-decision structured predictions (e.g., parsing) in intermediate layers. SPIGOT requires no marginal inference, unlike structured attention networks (Kim et al., 2017) and some reinforcement learning-inspired solutions (Yogatama et al., 2017). Like so-called straight-through estimators (Hinton, 2012), SPIGOT defines gradient-like quantities associated with intermediate nondifferentiable operations, allowing backpropagation before and after them; SPIGOT's proxy aims to ensure that, after a parameter update, the intermediate structure will remain well-formed.

We experiment on two structured NLP pipelines: syntactic-then-semantic dependency parsing, and semantic parsing followed by sentiment classification. We show that training with SPIGOT leads to a larger improvement on the downstream task than a modularly-trained pipeline, the straight-through estimator, and structured attention, reaching a new state of the art on semantic dependency parsing.

1 Introduction

Learning methods for natural language processing are increasingly dominated by end-to-end differentiable functions that can be trained using gradient-based optimization. Yet traditional NLP often assumed modular stages of processing that formed a pipeline; e.g., text was tokenized, then tagged with parts of speech, then parsed into a phrase-structure or dependency tree, then semantically analyzed. Pipelines, which make "hard" (i.e., discrete) decisions at each stage, appear to be incompatible with neural learning, leading many researchers to abandon earlier-stage processing.

Inspired by findings that continue to see benefit from various kinds of linguistic or domain-specific preprocessing (He et al., 2017; Oepen et al., 2017; Ji and Smith, 2017), we argue that pipelines can be treated as layers in neural architectures for NLP tasks. Several solutions are readily available:

- Reinforcement learning (most notably the REINFORCE algorithm; Williams, 1992), and **structured attention** (SA; Kim et al., 2017). These methods replace argmax with a sampling or marginalization operation. We note two potential downsides of these approaches: (i) not all argmax-able operations have corresponding sampling or marginalization methods that are efficient, and (ii) inspection of intermediate outputs, which could benefit error analysis and system improvement, is more straightforward for hard decisions than for posteriors.
- The **straight-through estimator** (STE; Hinton, 2012) treats discrete decisions as if they were differentiable and simply passes through gradients. While fast and surprisingly effective, it ignores *constraints* on the argmax problem, such as the requirement that every word has exactly one syntactic parent. We will find, experimentally, that the quality of intermediate representations degrades substantially under STE.

This paper introduces a new method, the **structured projection of intermediate gradients** optimization technique (SPIGOT; §2), which defines a proxy for the gradient of a loss function with respect to the input to argmax. Unlike STE's gradient proxy, SPIGOT aims to respect the constraints in the argmax problem. SPIGOT can be applied with any intermediate layer that is expressible as a constrained maximization problem, and whose feasible set can be projected onto. We show empirically that SPIGOT works even when the maximization and the projection are done approximately.

We offer two concrete architectures that employ structured argmax as an intermediate layer: semantic parsing with syntactic parsing in the middle, and sentiment analysis with semantic parsing in the middle (§3). These architectures are trained using a joint objective, with one part using data for the intermediate task, and the other using data for the end task. The datasets are not assumed to overlap at all, but the parameters for the intermediate task are affected by both parts of the training data.

Our experiments (§4) show that our architecture improves over a state-of-the-art semantic dependency parser, and that SPIGOT offers stronger performance than a pipeline, SA, and STE. On sentiment classification, we show that semantic parsing offers improvement over a BiLSTM, more so with SPIGOT than with alternatives. Our analysis considers how the behavior of the intermediate parser is affected by the end task (§5). Our code is open-source and available at `https://github.com/Noahs-ARK/SPIGOT`.

2 Method

Our aim is to allow a (structured) argmax layer in a neural network to be treated almost like any other differentiable function. This would allow us to place, for example, a syntactic parser in the middle of a neural network, so that the forward calculation simply calls the parser and passes the parse tree to the next layer, which might derive syntactic features for the next stage of processing.

The challenge is in the *backward* computation, which is key to learning with standard gradient-based methods. When its output is discrete as we assume here, argmax is a piecewise constant function. At every point, its gradient is either zero or undefined. So instead of using the true gradient, we will introduce a *proxy* for the gradient of the loss function with respect to the inputs to argmax, allowing backpropagation to proceed through the argmax layer. Our proxy is designed as an improvement to earlier methods (discussed below) that completely ignore constraints on the argmax operation. It accomplishes this through a projection of the gradients.

We first lay out notation, and then briefly review max-decoding and its relaxation (§2.1). We define SPIGOT in §2.2, and show how to use it to backpropagate through NLP pipelines in §2.3.

Notation. Our discussion centers around two tasks: a structured *intermediate* task followed by an *end* task, where the latter considers the outputs of the former (e.g., syntactic-then-semantic parsing). Inputs are denoted as $\mathbf{x}$, and end task outputs as $\mathbf{y}$. We use $\mathbf{z}$ to denote intermediate structures derived from $\mathbf{x}$. We will often refer to the intermediate task as "decoding", in the structured prediction sense. It seeks an output $\hat{\mathbf{z}} = \operatorname{argmax}_{\mathbf{z}\in\mathcal{Z}} S$ from the feasible set $\mathcal{Z}$, maximizing a (learned, parameterized) scoring function S for the structured intermediate task. L denotes the loss of the end task, which may or may not also involve structured predictions. We use $\Delta^{k-1} = \{\mathbf{p} \in \mathbb{R}^k \mid \mathbf{1}^\top\mathbf{p} = 1, \mathbf{p} \geq \mathbf{0}\}$ to denote the $(k-1)$-dimensional simplex. We denote the domain of binary variables as $\mathbb{B} = \{0, 1\}$, and the unit interval as $\mathbb{U} = [0, 1]$. By projection of a vector $\mathbf{v}$ onto a set $\mathcal{A}$, we mean the closest point in $\mathcal{A}$ to $\mathbf{v}$, measured by Euclidean distance: $\operatorname{proj}_{\mathcal{A}}(\mathbf{v}) = \operatorname{argmin}_{\mathbf{v}'\in\mathcal{A}} \|\mathbf{v}' - \mathbf{v}\|_2$.

2.1 Relaxed Decoding

Decoding problems are typically decomposed into a collection of "parts", such as arcs in a dependency tree or graph. In such a setup, each element of $\mathbf{z}$, z_i, corresponds to one possible part, and z_i takes a boolean value to indicate whether the part is included in the output structure. The scoring function S is assumed to decompose into a vector $\mathbf{s}(\mathbf{x})$ of part-local, input-specific scores:

$$\hat{\mathbf{z}} = \operatorname*{argmax}_{\mathbf{z}\in\mathcal{Z}} S(\mathbf{x}, \mathbf{z}) = \operatorname*{argmax}_{\mathbf{z}\in\mathcal{Z}} \mathbf{z}^\top\mathbf{s}(\mathbf{x}) \quad (1)$$

In the following, we drop $\mathbf{s}$'s dependence on $\mathbf{x}$ for clarity.

In many NLP problems, the output space $\mathcal{Z}$ can be specified by linear constraints (Roth and Yih, 2004):

$$\mathbf{A}\begin{bmatrix}\mathbf{z}\\ \boldsymbol{\psi}\end{bmatrix} \leq \mathbf{b}, \quad (2)$$

where $\boldsymbol{\psi}$ are auxiliary variables (also scoped by argmax), together with integer constraints (typically, each $z_i \in \mathbb{B}$).

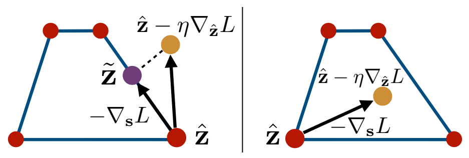

$\hat{\mathbf{z}} - \eta\nabla_{\hat{\mathbf{z}}}L$
$\tilde{\mathbf{z}}$
$-\nabla_{\mathbf{s}}L$
$\hat{\mathbf{z}}$
$\hat{\mathbf{z}} - \eta\nabla_{\hat{\mathbf{z}}}L$
$\hat{\mathbf{z}}$
$-\nabla_{\mathbf{s}}L$

Figure 1: The original feasible set $\mathcal{Z}$ (red vertices), is relaxed into a convex polytope $\mathcal{P}$ (the area encompassed by blue edges). Left: making a gradient update to $\hat{\mathbf{z}}$ makes it step outside the polytope, and it is projected back to $\mathcal{P}$, resulting in the projected point $\tilde{\mathbf{z}}$. $\nabla_{\mathbf{s}}L$ is then along the edge. Right: updating $\hat{\mathbf{z}}$ keeps it within $\mathcal{P}$, and thus $\nabla_{\mathbf{s}}L = \eta\nabla_{\hat{\mathbf{z}}}L$.

The problem in Equation 1 can be NP-complete in general, so the $\{0, 1\}$ constraints are often relaxed to $[0, 1]$ to make decoding tractable (Martins et al., 2009). Then the discrete combinatorial problem over $\mathcal{Z}$ is transformed into the optimization of a linear objective over a convex polytope $\mathcal{P}=\{\mathbf{p} \in \mathbb{R}^d | \mathbf{A}\mathbf{p} \leq \mathbf{b}\}$, which is solvable in polynomial time (Bertsimas and Tsitsiklis, 1997). This is not necessary in some cases, where the argmax can be solved exactly with dynamic programming.

2.2 From STE to SPIGOT

We now view structured argmax as an activation function that takes a vector of input-specific part-scores $\mathbf{s}$ and outputs a solution $\hat{\mathbf{z}}$. For backpropagation, to calculate gradients for parameters of $\mathbf{s}$, the chain rule defines:

$$\nabla_{\mathbf{s}}L = \mathbf{J}\,\nabla_{\hat{\mathbf{z}}}L, \tag{3}$$

where the Jacobian matrix $\mathbf{J} = \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{s}}$ contains the derivative of each element of $\hat{\mathbf{z}}$ with respect to each element of $\mathbf{s}$. Unfortunately, argmax is a piecewise constant function, so its Jacobian is either zero (almost everywhere) or undefined (in the case of ties).

One solution, taken in *structured attention*, is to replace the argmax with marginal inference and a softmax function, so that $\hat{\mathbf{z}}$ encodes probability distributions over parts (Kim et al., 2017; Liu and Lapata, 2018). As discussed in §1, there are two reasons to avoid this modification. Softmax can only be used when marginal inference is feasible, by sum-product algorithms for example (Eisner, 2016; Friesen and Domingos, 2016); in general marginal inference can be #P-complete. Further, a soft intermediate layer will be less amenable to inspection by anyone wishing to understand and improve the model.

In another line of work, argmax is augmented with a strongly-convex penalty on the solutions (Martins and Astudillo, 2016; Amos and Kolter, 2017; Niculae and Blondel, 2017; Niculae et al., 2018; Mensch and Blondel, 2018). However, their approaches require solving a relaxation even when exact decoding is tractable. Also, the penalty will bias the solutions found by the decoder, which may be an undesirable conflation of computational and modeling concerns.

A simpler solution is the STE method (Hinton, 2012), which replaces the Jacobian matrix in Equation 3 by the identity matrix. This method has been demonstrated to work well when used to "backpropagate" through hard threshold functions (Bengio et al., 2013; Friesen and Domingos, 2018) and categorical random variables (Jang et al., 2016; Choi et al., 2017).

Consider for a moment what we would do if $\hat{\mathbf{z}}$ were a vector of parameters, rather than intermediate predictions. In this case, we are seeking points in $\mathcal{Z}$ that minimize L; denote that set of minimizers by $\mathcal{Z}^*$. Given $\nabla_{\hat{\mathbf{z}}}L$ and step size η, we would update $\hat{\mathbf{z}}$ to be $\hat{\mathbf{z}} - \eta\nabla_{\hat{\mathbf{z}}}L$. This update, however, might not return a value in the feasible set $\mathcal{Z}$, or even (if we are using a linear relaxation) the relaxed set $\mathcal{P}$.

SPIGOT therefore introduces a *projection* step that aims to keep the "updated" $\hat{\mathbf{z}}$ in the feasible set. Of course, we do not directly update $\hat{\mathbf{z}}$; we continue backpropagation through $\mathbf{s}$ and onward to the parameters. But the projection step nonetheless alters the parameter updates in the way that our proxy for "$\nabla_{\mathbf{s}}L$" is defined.

The procedure is defined as follows:

$$\hat{\mathbf{p}} = \hat{\mathbf{z}} - \eta\nabla_{\hat{\mathbf{z}}}L, \tag{4a}$$
$$\tilde{\mathbf{z}} = \mathrm{proj}_{\mathcal{P}}(\hat{\mathbf{p}}), \tag{4b}$$
$$\nabla_{\mathbf{s}}L \triangleq \hat{\mathbf{z}} - \tilde{\mathbf{z}}. \tag{4c}$$

First, the method makes an "update" to $\hat{\mathbf{z}}$ as if it contained parameters (Equation 4a), letting $\hat{\mathbf{p}}$ denote the new value. Next, $\hat{\mathbf{p}}$ is projected back onto the (relaxed) feasible set (Equation 4b), yielding a feasible new value $\tilde{\mathbf{z}}$. Finally, the gradients with respect to $\mathbf{s}$ are computed by Equation 4c.

Due to the convexity of $\mathcal{P}$, the projected point $\tilde{\mathbf{z}}$ will always be unique, and is guaranteed to be no farther than $\hat{\mathbf{p}}$ from any point in $\mathcal{Z}^*$ (Luenberger and Ye, 2015).[1] Compared to STE, SPIGOT in-

[1]Note that this property follows from $\mathcal{P}$'s convexity, and we do not assume the convexity of L.

volves a projection and limits $\nabla_{\mathbf{s}} L$ to a smaller space to satisfy constraints. See Figure 1 for an illustration.

When efficient exact solutions (such as dynamic programming) are available, they can be used. Yet, we note that SPIGOT does not assume the argmax operation is solved exactly.

2.3 Backpropagation through Pipelines

Using SPIGOT, we now devise an algorithm to "backpropagate" through NLP pipelines. In these pipelines, an intermediate task's output is fed into an end task for use as features. The parameters of the complete model are divided into two parts: denote the parameters of the intermediate task model by $\boldsymbol{\phi}$ (used to calculate $\mathbf{s}$), and those in the end task model as $\boldsymbol{\theta}$.[2] As introduced earlier, the end-task loss function to be minimized is L, which depends on both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$.

Algorithm 1 describes the forward and backward computations. It takes an end task training pair $\langle \mathbf{x}, \mathbf{y} \rangle$, along with the intermediate task's feasible set $\mathcal{Z}$, which is determined by $\mathbf{x}$. It first runs the intermediate model and decodes to get intermediate structure $\hat{\mathbf{z}}$, just as in a standard pipeline. Then forward propagation is continued into the end-task model to compute loss L, using $\hat{\mathbf{z}}$ to define input features. Backpropagation in the end-task model computes $\nabla_{\boldsymbol{\theta}} L$ and $\nabla_{\hat{\mathbf{z}}} L$, and $\nabla_{\mathbf{s}} L$ is then constructed using Equations 4. Backpropagation then continues into the intermediate model, computing $\nabla_{\boldsymbol{\phi}} L$.

Due to its flexibility, SPIGOT is applicable to many training scenarios. When there is no $\langle \mathbf{x}, \mathbf{z} \rangle$ training data for the intermediate task, SPIGOT can be used to induce latent structures for the end-task (Yogatama et al., 2017; Kim et al., 2017; Choi et al., 2017, *inter alia*). When intermediate-task training data *is* available, one can use SPIGOT to adopt joint learning by minimizing an interpolation of L (on end-task data $\langle \mathbf{x}, \mathbf{y} \rangle$) and an intermediate-task loss function $\widetilde{L}$ (on intermediate task data $\langle \mathbf{x}, \mathbf{z} \rangle$). This is the setting in our experiments; note that we do not assume any overlap in the training examples for the two tasks.

3 Solving the Projections

In this section we discuss how to compute approximate projections for the two intermediate tasks considered in this work, arc-factored unlabeled dependency parsing and first-order semantic dependency parsing.

[2] Nothing prohibits tying across pre-argmax parameters and post-argmax parameters; this separation is notationally convenient but not at all necessary.

Algorithm 1 Forward and backward computation with SPIGOT.

```
1:  procedure SPIGOT($\mathbf{x}, \mathbf{y}, \mathcal{Z}$)
2:      Construct $\mathbf{A}, \mathbf{b}$ such that $\mathcal{Z} = \{\mathbf{p} \in \mathbb{Z}^d \mid \mathbf{Ap} \leq \mathbf{b}\}$
3:      $\mathcal{P} \leftarrow \{\mathbf{p} \in \mathbb{R}^d \mid \mathbf{Ap} \leq \mathbf{b}\}$    ▷ Relaxation
4:      Forwardprop and compute $\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x})$
5:      $\hat{\mathbf{z}} \leftarrow \mathrm{argmax}_{\mathbf{z} \in \mathcal{Z}}\, \mathbf{z}^\top \mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x})$    ▷ Intermediate decoding
6:      Forwardprop and compute $L$ given $\mathbf{x}$, $\mathbf{y}$, and $\hat{\mathbf{z}}$
7:      Backprop and compute $\nabla_{\boldsymbol{\theta}} L$ and $\nabla_{\hat{\mathbf{z}}} L$
8:      $\tilde{\mathbf{z}} \leftarrow \mathrm{proj}_{\mathcal{P}}(\hat{\mathbf{z}} - \eta \nabla_{\hat{\mathbf{z}}} L)$    ▷ Projection
9:      $\nabla_{\mathbf{s}} L \leftarrow \hat{\mathbf{z}} - \tilde{\mathbf{z}}$
10:     Backprop and compute $\nabla_{\boldsymbol{\phi}} L$
11: end procedure
```

In early experiments we observe that for both tasks, projecting with respect to *all* constraints of their original formulations using a generic quadratic program solver was prohibitively slow. Therefore, we construct relaxed polytopes by considering only a subset of the constraints.[3] The projection then decomposes into a series of singly constrained quadratic programs (QP), each of which can be efficiently solved in linear time.

The two approximate projections discussed here are used in backpropagation only. In the forward pass, we solve the decoding problem using the models' original decoding algorithms.

Arc-factored unlabeled dependency parsing. For unlabeled dependency trees, we impose $[0, 1]$ constraints and single-headedness constraints.[4]

Formally, given a length-n input sentence, excluding self-loops, an arc-factored parser considers $d = n(n-1)$ candidate arcs. Let $i{\to}j$ denote an arc from the ith token to the jth, and $\sigma(i{\to}j)$ denote its index. We construct the relaxed feasible set by:

$$\mathcal{P}_{\text{DEP}} = \left\{ \mathbf{p} \in \mathbb{U}^d \;\middle|\; \sum_{i \neq j} p_{\sigma(i \to j)} = 1, \forall j \right\}, \quad (5)$$

i.e., we consider each token j individually, and force single-headedness by constraining the number of arcs incoming to j to sum to 1. Algorithm 2 summarizes the procedure to project onto $\mathcal{P}_{\text{DEP}}$.

[3] A parallel work introduces an active-set algorithm to solve the same class of quadratic programs (Niculae et al., 2018). It might be an efficient approach to solve the projections in Equation 4b, which we leave to future work.

[4] It requires $O(n^2)$ auxiliary variables and $O(n^3)$ additional constraints to ensure well-formed tree structures (Martins et al., 2013).

Line 3 forms a singly constrained QP, and can be solved in $O(n)$ time (Brucker, 1984).

Algorithm 2 Projection onto the relaxed polytope $\mathcal{P}_{\text{DEP}}$ for dependency tree structures. Let bold $\boldsymbol{\sigma}(\cdot{\to}j)$ denote the index set of arcs incoming to j. For a vector $\mathbf{v}$, we use $\mathbf{v}_{\boldsymbol{\sigma}(\cdot\to j)}$ to denote vector $[v_k]_{k\in\boldsymbol{\sigma}(\cdot\to j)}$.

```
1: procedure DEPPROJ($\hat{\mathbf{p}}$)
2:     for $j = 1, 2, \ldots, n$ do
3:         $\tilde{\mathbf{z}}_{\boldsymbol{\sigma}(\cdot\to j)} \leftarrow \text{proj}_{\triangle^{n-2}}\left(\hat{\mathbf{p}}_{\boldsymbol{\sigma}(\cdot\to j)}\right)$
4:     end for
5:     return $\tilde{\mathbf{z}}$
6: end procedure
```

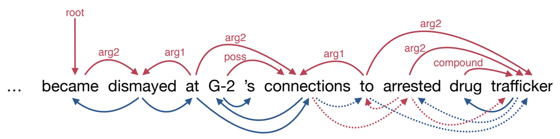

root
arg2
arg1
arg2
poss
arg1
arg2
arg2
compound
… became dismayed at G-2 's connections to arrested drug trafficker

Figure 2: A development instance annotated with both gold DM semantic dependency graph (red arcs on the top), and gold syntactic dependency tree (blue arcs at the bottom). A pretrained syntactic parser predicts the same tree as the gold; the semantic parser backpropagates into the intermediate syntactic parser, and changes the dashed blue arcs into dashed red arcs (§5).

First-order semantic dependency parsing. Semantic dependency parsing uses labeled bilexical dependencies to represent sentence-level semantics (Oepen et al., 2014, 2015, 2016). Each dependency is represented by a labeled directed arc from a head token to a modifier token, where the arc label encodes broadly applicable semantic relations. Figure 2 diagrams a semantic graph from the DELPH-IN MRS-derived dependencies (DM), together with a syntactic tree.

We use a state-of-the-art semantic dependency parser (Peng et al., 2017) that considers three types of parts: heads, unlabeled arcs, and labeled arcs. Let $\sigma(i \xrightarrow{\ell} j)$ denote the index of the arc from i to j with semantic role ℓ. In addition to $[0, 1]$ constraints, we constrain that the predictions for labeled arcs sum to the prediction of their associated unlabeled arc:

$$\mathcal{P}_{\text{SDP}}\left\{\mathbf{p}\in\mathbb{U}^d \;\middle|\; \sum_{\ell} p_{\sigma(i\xrightarrow{\ell}j)} = p_{\sigma(i\to j)}, \forall i\neq j\right\}. \tag{6}$$

This ensures that exactly one label is predicted if and only if its arc is present. The projection onto $\mathcal{P}_{\text{SDP}}$ can be solved similarly to Algorithm 2. We drop the determinism constraint imposed by Peng et al. (2017) in the backward computation.

4 Experiments

We empirically evaluate our method with two sets of experiments: using syntactic tree structures in semantic dependency parsing, and using semantic dependency graphs in sentiment classification.

4.1 Syntactic-then-Semantic Parsing

In this experiment we consider an intermediate syntactic parsing task, followed by semantic dependency parsing as the end task. We first briefly review the neural network architectures for the two models (§4.1.1), and then introduce the datasets (§4.1.2) and baselines (§4.1.3).

4.1.1 Architectures

Syntactic dependency parser. For intermediate syntactic dependencies, we use the unlabeled arc-factored parser of Kiperwasser and Goldberg (2016). It uses bidirectional LSTMs (BiLSTM) to encode the input, followed by a multilayer-perceptron (MLP) to score each potential dependency. One notable modification is that we replace their use of Chu-Liu/Edmonds' algorithm (Chu and Liu, 1965; Edmonds, 1967) with the Eisner algorithm (Eisner, 1996, 2000), since our dataset is in English and mostly projective.

Semantic dependency parser. We use the basic model of Peng et al. (2017) (denoted as NeurboParser) as the end model. It is a first-order parser, and uses local factors for heads, unlabeled arcs, and labeled arcs. NeurboParser does not use syntax. It first encodes an input sentence with a two-layer BiLSTM, and then computes part scores with two-layer tanh-MLPs. Inference is conducted with AD^3 (Martins et al., 2015). To add syntactic features to NeurboParser, we concatenate a token's contextualized representation to that of its syntactic head, predicted by the intermediate parser. Formally, given length-n input sentence, we first run a BiLSTM. We use the concatenation of the two hidden representations $\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]$ at each position j as the contextualized token representations. We then concatenate

$\mathbf{h}_j$ with the representation of its head $\mathbf{h}_{\text{HEAD}(j)}$ by

$$\widetilde{\mathbf{h}}_j = [\mathbf{h}_j; \mathbf{h}_{\text{HEAD}(j)}] = \left[\mathbf{h}_j; \sum_{i \neq j} \hat{z}_{\sigma(i \to j)}\, \mathbf{h}_i\right], \tag{7}$$

where $\hat{\mathbf{z}} \in \mathbb{B}^{n(n-1)}$ is a binary encoding of the tree structure predicted by by the intermediate parser. We then use $\widetilde{\mathbf{h}}_j$ anywhere $\mathbf{h}_j$ would have been used in NEURBOPARSER. In backpropagation, we compute $\nabla_{\hat{\mathbf{z}}} L$ with an automatic differentiation toolkit (DyNet; Neubig et al., 2017).

We note that this approach can be generalized to convolutional neural networks over graphs (Mou et al., 2015; Duvenaud et al., 2015; Kipf and Welling, 2017, *inter alia*), recurrent neural networks along paths (Xu et al., 2015; Roth and Lapata, 2016, *inter alia*) or dependency trees (Tai et al., 2015). We choose to use concatenations to control the model's complexity, and thus to better understand which parts of the model work.

We refer the readers to Kiperwasser and Goldberg (2016) and Peng et al. (2017) for further details of the parsing models.

Training procedure. Following previous work, we minimize structured hinge loss (Tsochantaridis et al., 2004) for both models. We jointly train both models from scratch, by randomly sampling an instance from the union of their training data at each step. In order to isolate the effect of backpropagation, we do not share any parameters between the two models.[5] Implementation details are summarized in the supplementary materials.

4.1.2 Datasets

- For semantic dependencies, we use the English dataset from SemEval 2015 Task 18 (Oepen et al., 2015). Among the three formalisms provided by the shared task, we consider DELPH-IN MRS-derived dependencies (DM) and Prague Semantic Dependencies (PSD).[6] It includes §00–19 of the WSJ corpus as training data, §20 and §21 for development and in-domain test data, resulting in a 33,961/1,692/1,410 train/dev./test split, and 1,849 out-of-domain test instances from the Brown corpus.[7]
- For syntactic dependencies, we use the Stanford Dependency (de Marneffe and Manning, 2008) conversion of the the Penn Treebank WSJ portion (Marcus et al., 1993). To avoid data leak, we depart from standard split and use §20 and §21 as development and test data, and the remaining sections as training data. The number of training/dev./test instances is 40,265/2,012/1,671.

	DM		PSD	
Model	UF	LF	UF	LF
NEURBOPARSER	–	89.4	–	77.6
FREDA3	–	90.4	–	78.5
PIPELINE	91.8	90.8	88.4	78.1
SA	91.6	90.6	87.9	78.1
STE	92.0	91.1	**88.9**	**78.9**
SPIGOT	**92.4**	**91.6**	88.6	**78.9**

(a) F_1 on in-domain test set.

	DM		PSD	
Model	UF	LF	UF	LF
NEURBOPARSER	–	84.5	–	75.3
FREDA3	–	85.3	–	76.4
PIPELINE	87.4	85.8	85.5	75.6
SA	87.3	85.6	84.9	75.9
STE	87.7	86.4	**85.8**	76.6
SPIGOT	**87.9**	**86.7**	85.5	**77.1**

(b) F_1 on out-of-domain test set.

Table 1: Semantic dependency parsing performance in both unlabeled (UF) and labeled (LF) F_1 scores. Bold font indicates the best performance. Peng et al. (2017) does not report UF.

4.1.3 Baselines

We compare to the following baselines:

- A pipelined system (PIPELINE). The pretrained parser achieves 92.9 test unlabeled attachment score (UAS).[8]

[5] Parameter sharing has proved successful in many related tasks (Collobert and Weston, 2008; Søgaard and Goldberg, 2016; Ammar et al., 2016; Swayamdipta et al., 2016, 2017, *inter alia*), and could be easily combined with our approach.

[6] We drop the third (PAS) because its structure is highly predictable from parts-of-speech, making it less interesting.

[7] The organizers remove, e.g., instances with cyclic graphs, and thus only a subset of the WSJ corpus is included. See Oepen et al. (2015) for details.

[8] Note that this number is not comparable to the parsing literature due to the different split. As a sanity check, we found in preliminary experiments that the same parser archi-

- Structured attention networks (SA; Kim et al., 2017). We use the inside-outside algorithm (Baker, 1979) to populate $\mathbf{z}$ with arcs' marginal probabilities, use log-loss as the objective in training the intermediate parser.
- The straight-through estimator (STE; Hinton, 2012), introduced in §2.2.

4.1.4 Empirical Results

Table 1 compares the semantic dependency parsing performance of SPIGOT to all five baselines. FREDA3 (Peng et al., 2017) is a state-of-the-art variant of NEURBOPARSER that is trained using multitask learning to jointly predict three different semantic dependency graph formalisms. Like the basic NEURBOPARSER model that we build from, FREDA3 does not use any syntax. Strong DM performance is achieved in a more recent work by using joint learning and an ensemble (Peng et al., 2018), which is beyond fair comparisons to the models discussed here.

We found that using syntactic information improves semantic parsing performance: using pipelined syntactic head features brings 0.5–1.4% absolute labeled F_1 improvement to NEURBOPARSER. Such improvements are smaller compared to previous works, where dependency path and syntactic relation features are included (Almeida and Martins, 2015; Ribeyre et al., 2015; Zhang et al., 2016), indicating the potential to get better performance by using more syntactic information, which we leave to future work.

Both STE and SPIGOT use hard syntactic features. By allowing backpropation into the intermediate syntactic parser, they both consistently outperform PIPELINE. On the other hand, when marginal syntactic tree structures are used, SA outperforms PIPELINE only on the out-of-domain PSD test set, and improvements under other cases are not observed.

Compared to STE, SPIGOT outperforms STE on DM by more than 0.3% absolute labeled F_1, both in-domain and out-of-domain. For PSD, SPIGOT achieves similar performance to STE on in-domain test set, but has a 0.5% absolute labeled F_1 improvement on out-of-domain data, where syntactic parsing is less accurate.

tecture achieves 93.5 UAS when trained and evaluated with the standard split, close to the results reported by Kiperwasser and Goldberg (2016).

4.2 Semantic Dependencies for Sentiment Classification

Our second experiment uses semantic dependency graphs to improve sentiment classification performance. We are not aware of any efficient algorithm that solves marginal inference for semantic dependency graphs under determinism constraints, so we do not include a comparison to SA.

4.2.1 Architectures

Here we use NEURBOPARSER as the intermediate model, as described in §4.1.1, but with no syntactic enhancements.

Sentiment classifier. We first introduce a baseline that does not use any structural information. It learns a one-layer BiLSTM to encode the input sentence, and then feeds the sum of all hidden states into a two-layer ReLU-MLP.

To use semantic dependency features, we concatenate a word's BiLSTM-encoded representation to the averaged representation of its heads, together with the corresponding semantic roles, similarly to that in Equation 7.[9] Then the concatenation is fed into an affine transformation followed by a ReLU activation. The rest of the model is kept the same as the BiLSTM baseline.

Training procedure. We use structured hinge loss to train the semantic dependency parser, and log-loss for the sentiment classifier. Due to the discrepancy in the training data size of the two tasks (33K vs. 7K), we pre-train a semantic dependency parser, and then adopt joint training together with the classifier. In the joint training stage, we randomly sample 20% of the semantic dependency training instances each epoch. Implementations are detailed in the supplementary materials.

4.2.2 Datasets

For semantic dependencies, we use the DM dataset introduced in §4.1.2.

We consider a binary classification task using the Stanford Sentiment Treebank (Socher et al., 2013). It consists of roughly 10K movie review sentences from Rotten Tomatoes. The full dataset includes a rating on a scale from 1 to 5 for each constituent (including the full sentences), resulting in more than 200K instances. Following previous work (Iyyer et al., 2015), we only use full-sentence

[9]In a well-formed semantic dependency graph, a token may have multiple heads. Therefore we use average instead of the sum in Equation 7.

Model	Accuracy (%)
BiLSTM	84.8
PIPELINE	85.7
STE	85.4
SPIGOT	**86.3**

Table 2: Test accuracy of sentiment classification on Stanford Sentiment Treebank. Bold font indicates the best performance.

instances, with neutral instances excluded (3s) and the remaining four rating levels converted to binary "positive" or "negative" labels. This results in a 6,920/872/1,821 train/dev./test split.

4.2.3 Empirical Results

Table 2 compares our SPIGOT method to three baselines. Pipelined semantic dependency predictions brings 0.9% absolute improvement in classification accuracy, and SPIGOT outperforms all baselines. In this task STE achieves slightly worse performance than a fixed pre-trained PIPELINE.

5 Analysis

We examine here how the intermediate model is affected by the end-task training signal. Is the end-task signal able to "overrule" intermediate predictions?

We use the syntactic-then-semantic parsing model (§4.1) as a case study. Table 3 compares a pipelined system to one jointly trained using SPIGOT. We consider the development set instances where both syntactic and semantic annotations are available, and partition them based on whether the two systems' syntactic predictions agree (SAME), or not (DIFF). The second group includes sentences with much lower syntactic parsing accuracy (91.3 vs. 97.4 UAS), and SPIGOT further reduces this to 89.6. Even though these changes hurt syntactic parsing accuracy, they lead to a 1.1% absolute gain in labeled F_1 for semantic parsing. Furthermore, SPIGOT has an overall less detrimental effect on the intermediate parser than STE: using SPIGOT, intermediate dev. parsing UAS drops to 92.5 from the 92.9 pipelined performance, while STE reduces it to 91.8.

We then take a detailed look and categorize the changes in intermediate trees by their correlations with the semantic graphs. Specifically, when a modifier m's head is changed from h to h' in the tree, we consider three cases: (a) h' is a head of m in the semantic graph; (b) h' is a modifier of m in the semantic graph; (c) h is the modifier of m in the semantic graph. The first two reflect modifications to the syntactic parse that rearrange semantically linked words to be neighbors. Under (c), the semantic parser removes a syntactic dependency that reverses the direction of a semantic dependency. These cases account for 17.6%, 10.9%, and 12.8%, respectively (41.2% combined) of the total changes. Making these changes, of course, is complicated, since they often require *other* modifications to maintain well-formedness of the tree. Figure 2 gives an example.

Split	# Sent.	Model	UAS	DM
SAME	1011	PIPELINE	97.4	94.0
		SPIGOT	97.4	94.3
DIFF	681	PIPELINE	91.3	88.1
		SPIGOT	89.6	89.2

Table 3: Syntactic parsing performance (in unlabeled attachment score, UAS) and DM semantic parsing performance (in labeled F_1) on different groups of the development data. Both systems predict the same syntactic parses for instances from SAME, and they disagree on instances from DIFF (§5).

6 Related Work

Joint learning in NLP pipelines. To avoid cascading errors, much effort has been devoted to joint decoding in NLP pipelines (Habash and Rambow, 2005; Cohen and Smith, 2007; Goldberg and Tsarfaty, 2008; Lewis et al., 2015; Zhang et al., 2015, *inter alia*). However, joint inference can sometimes be prohibitively expensive. Recent advances in representation learning facilitate exploration in the joint learning of multiple tasks by sharing parameters (Collobert and Weston, 2008; Blitzer et al., 2006; Finkel and Manning, 2010; Zhang and Weiss, 2016; Hashimoto et al., 2017, *inter alia*).

Differentiable optimization. Gould et al. (2016) review the generic approaches to differentiation in bi-level optimization (Bard, 2010; Kunisch and Pock, 2013). Amos and Kolter (2017) extend their efforts to a class of subdifferentiable quadratic programs. However, they both require that the intermediate objective has an invertible Hessian, limiting their application in NLP. In another line of work, the steps of a gradient-based optimization procedure are unrolled into a single computation graph (Stoyanov et al., 2011; Domke, 2012; Goodfellow et al., 2013; Brakel et al., 2013). This comes at a high computational cost due to the second-order derivative computation during backpropagation. Moreover, constrained optimization problems (like many NLP problems) often require projection steps within the procedure, which can be difficult to differentiate through (Belanger and McCallum, 2016; Belanger et al., 2017).

7 Conclusion

We presented SPIGOT, a novel approach to backpropagating through neural network architectures that include discrete structured decisions in intermediate layers. SPIGOT devises a proxy for the gradients with respect to argmax's inputs, employing a projection that aims to respect the constraints in the intermediate task. We empirically evaluate our method with two architectures: a semantic parser with an intermediate syntactic parser, and a sentiment classifier with an intermediate semantic parser. Experiments show that SPIGOT achieves stronger performance than baselines under both settings, and outperforms state-of-the-art systems on semantic dependency parsing. Our implementation is available at `https://github.com/Noahs-ARK/SPIGOT`.

Acknowledgments

We thank the ARK, Julian Michael, Minjoon Seo, Eunsol Choi, and Maxwell Forbes for their helpful comments on an earlier version of this work, and the anonymous reviewers for their valuable feedback. This work was supported in part by NSF grant IIS-1562364.

References

Mariana S. C. Almeida and André F. T. Martins. 2015. Lisbon: Evaluating TurboSemanticParser on multiple languages and out-of-domain data. In *Proc. of SemEval*.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *TACL* 4:431–444.

Brandon Amos and J. Zico Kolter. 2017. OptNet: Differentiable optimization as a layer in neural networks. In *Proc. of ICML*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*.

Jonathan F. Bard. 2010. *Practical Bilevel Optimization: Algorithms and Applications*. Springer.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proc. of ICML*.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *Proc. of ICML*.

Yoshua Bengio, Nicholas Lonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432.

Dimitris Bertsimas and John Tsitsiklis. 1997. *Introduction to Linear Optimization*. Athena Scientific.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*.

Philémon Brakel, Dirk Stroobandt, and Benjamin Schrauwen. 2013. Training energy-based models for time-series imputation. *Journal of Machine Learning Research* 14:2771–2797.

Peter Brucker. 1984. An $O(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters* 3(3):163 – 166.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-LSTMs. arXiv:1707.02786.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica* 14:1396–1400.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proc. of EMNLP*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

Justin Domke. 2012. Generic methods for optimization-based modeling. In *Proc. of AISTATS*.

David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proc. of NIPS*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards* 71B:233–240.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*, Springer Netherlands, pages 29–61.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop. In *Proceedings of the EMNLP Workshop on Structured Prediction for NLP*.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

Jenny Rose Finkel and Christopher D. Manning. 2010. Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data. In *Proc. of ACL*.

Abram L. Friesen and Pedro M. Domingos. 2016. The sum-product theorem: A foundation for learning tractable models. In *Proc. of ICML*.

Abram L. Friesen and Pedro M. Domingos. 2018. Deep learning as a mixed convex-combinatorial optimization problem. In *Proc. of ICLR*.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proc. of ACL*.

Ian Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Multi-prediction deep Boltzmann machines. In *Proc. of NIPS*.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. 2016. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. arXiv:1607.05447.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proc. ACL*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proc. of EMNLP*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proc. of ACL*.

Geoffrey Hinton. 2012. Neural networks for machine learning. *Coursera* video lectures.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with Gumbel-Softmax. arXiv:1611.01144.

Yangfeng Ji and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *Proc. of ACL*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proc. of ICLR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*.

Karl Kunisch and Thomas Pock. 2013. A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences* 6(2):938–983.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proc. of EMNLP*.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *TACL* 6:63–75.

David G. Luenberger and Yinyu Ye. 2015. *Linear and Nonlinear Programming*. Springer.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics* 19(2):313–330.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proc. of ICML*.

André F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. 2015. AD3: Alternating directions dual decomposition for map inference in graphical models. *Journal of Machine Learning Research* 16:495–545.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *Proc. of ICML*.

Arthur Mensch and Mathieu Blondel. 2018. Differentiable dynamic programming for structured prediction and attention. *arXiv:1802.03676* .

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proc. of EMNLP*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *Proc. of NIPS*.

Vlad Niculae, Andr F. T. Martins, Mathieu Blondel, and Claire Cardie. 2018. SparseMAP: Differentiable sparse structured inference. arXiv:1802.04223.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Stephan Oepen, Lilja vrelid, Jari Bjrne, Richard Johansson, Emanuele Lapponi, Filip Ginter, and Erik Velldal. 2017. The 2017 shared task on extrinsic parser evaluation. towards a reusable community infrastructure. In *Proc. of the 2017 Shared Task on Extrinsic Parser Evaluation*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*.

Corentin Ribeyre, Éric Villemonte De La Clergerie, and Djamé Seddah. 2015. Because syntax does matter: Improving predicate-argument structures parsing using syntactic features. In *Proc. of NAACL*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of NAACL*.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proc. of ACL*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. 2011. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proc. of AISTATS*.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proc. of CoNLL*.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. arXiv:1706.09528.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL*.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4):229–256.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. of EMNLP*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *Proc. of ICLR*.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *Proc. NAACL*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*.

Supplementary Materials: Backpropagating through Structured Argmax using a SPIGOT

Hao Peng◇ **Sam Thomson**♣ **Noah A. Smith**◇
◇ Paul G. Allen School of Computer Science & Engineering, University of Washington
♣ School of Computer Science, Carnegie Mellon University
{hapeng,nasmith}@cs.washington.edu, sthomson@cs.cmu.edu

1 Implementation Details

Our implementation is based on the DyNet toolkit.[1] We use part-of-speech tags and lemmas predicted by NLTK.[2]

1.1 Syntactic-then-Semantic Parsing Experiment

Each input token is represented as the concatenation a word embedding vector, a learned lemma vector, and a learned vector for part-of-speech, all updated during training. In joint training, we apply early-stopping based on semantic dependency parsing development performance (in labeled F_1). We do not use mini-batch. We set the step size η for SPIGOT to 1.

Semantic dependency parser. We use the pruning techniques in Martins and Almeida (2014), and replace their feature-rich model with neural networks (Peng et al., 2018). We observe that the number of parts surviving pruning is linear in the sentence length ($5.5\times$ on average), with $\sim$99% recall.

We do not deviate far from the hyperparameter setting in Peng et al. (2017), with the only exception being that we use 50-dimensional lemma and part-of-speech embeddings, instead of 25.

Syntactic dependency parser. For the max-margin syntactic parsers used in PIPELINE, STE, and SPIGOT, we use the hyperparameters reported in Kiperwasser and Goldberg (2016), but replace their 125-dimensional BiLSTMs with 200-dimensional ones, and use 50-dimensional POS embeddings, instead of 25. We anneal the learning rate at a rate of 0.5 every 5 epochs.

For the marginal syntactic parser in SA, we follow the use of Adam algorithm (Kingma and Ba, 2015), but set a smaller initial learning rate of 5×10^{-4}, annealed at a rate of 0.5 every 4 epochs. The rest of the hyperparameters stay the same as the max-margin parser.

[1]https://github.com/clab/dynet
[2]http://www.nltk.org/

Hyperparameter	Values
MLP dimension	$\{100, 150, 200, 250, 300\}$
BiLSTM dimension	$\{100, 150, 200, 250, 300\}$
Embedding dropout	$\{0.2, 0.3, 0.4, 0.5\}$
MLP dropout	$\{0.0, 0.1, 0.2, 0.3, 0.4\}$

Table 1: Hyperparameters explored in sentiment classification experiments.

1.2 Semantic Parsing and Sentiment Classification Experiment

The model is trained for up to 30 epochs in the joint training stage. We apply early-stopping based on sentiment classification development accuracy. For semantic dependency parser, we follow the hyperparameters described in §1.1.

Sentiment classifier. We use 300-dimensional GloVe (Pennington et al., 2014) to initialize word embeddings, fixed during training. We use a single-layer BiLSTM, followed by a two-layer ReLU-MLP. Dropout in word embeddings and MLPs is applied, but not in LSTMs. We use Adam algorithm (Kingma and Ba, 2015), and follow the default procedures by DyNet for optimizer settings and parameter initializations. An ℓ_2-penalty of 10^{-6} is applied to all weights. Learning rate is annealed at a rate of 0.5 every 5 epochs. We use mini-batches of 32, and clip the ℓ_2-norm of gradients to 5 (Graves, 2013). We set the step size η for SPIGOT to $\frac{5}{32}$. We explore the same set of hyperparameters based on development performance for all compared models, summarized in Table 1.

References

Alex Graves. 2013. Generating sequences with recurrent neural networks. arXiv:1308.0850.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A Turbo semantic parser with second order features. In *Proc. of SemEval*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

November 26, 2021

To whom it may concern:

I am writing to recommend Hao Peng for a tenure-track faculty or equivalent faculty position at your institution. Hao is a brilliant, independent researcher whose creative, high-impact research ideas and unique insights have produced a formidable body of groundbreaking research that is, in my view, among the most exciting work in NLP in the past few years. He'll be an outstanding addition to your department, and I recommend him as strongly as one possibly can.

My connection to the applicant: I have known Hao since early 2016, shortly before he joined the University of Washington's Ph.D. program in Computer Science & Engineering as my advisee. Hao was the only Ph.D. student joining my group that year, and I was extremely honored that he chose to come. At that time, he had already published four papers (as an undergraduate!) at top venues in natural language processing (ACL and EMNLP) and machine learning (ICML), and he came highly recommended by internship mentor Chin-Yew Lin from Microsoft Research and by Prof. Charles Sutton, with whom he had done a visiting student stint at the University of Edinburgh. Hao is the only advisee of mine (in more than fifteen years of amazing students) to have turned down a competing offer of admission from MIT.

The field of natural language processing (NLP), where Hao and I have found our intellectual homes, has undergone substantial changes in the time of Hao's Ph.D. By 2016, neural networks (branded as "deep learning") had emerged as a powerful tool for data-driven modeling of natural language and for building systems that work with text (e.g., automatic question answering, machine translation, constructing knowledge bases from unstructured text collections, and so on).

Structured prediction. A view in the research community, still held in 2016 but not as widely today, was that the inherent structure of linguistic utterances – including both their syntax (i.e., their adherence to rules of grammar implicitly known by native speakers of the language) and semantics (i.e., a mapping to some abstract representation of meaning in a world) – was central to the goals of NLP. There was great excitement about harnessing the powerful pattern recognition capabilities of neural networks to carry out the parsing of sentences, explicitly, into these syntactic and semantic structures. Structured prediction methods that work on real-world text have been a major focus in my lab, and this is where Hao's Ph.D. journey began.

Hao's first big result at UW was "Neurboparser." Working with senior Ph.D. student Sam Thomson (now at Semantic Machines), he combined the powerful structured inference technique AD^3 (developed by Andre Martins in our group a few years before) with a representation learner based on recurrent neural networks and achieved significant gains on semantic dependency parsing (across three formalisms derived from various flavors of minimal recursion semantics). This work is analogous to work on syntactic dependencies by Yoav Goldberg and colleagues; Hao and Sam went further in building a single parser that learned from data annotated according to *all three* semantic formalisms together. This is a beautiful and early example of the power of multitask (neural) learning – using differently annotated data to improve performance on all the tasks. Multitask learning, or more generally settings that involve heterogeneously annotated data (and perhaps also unannotated data) combined in various ways (such as pretraining), is now widely studied in our research community, including the notable "ELMo" result by Matt Peters et al., 2018, in which pretrained neural language models were shown to be an extremely effective way to leverage unannotated data for various NLP tasks. Hao's work was published at ACL

Noah A. Smith
Amazon Professor of Machine Learning
Box 352350 | Seattle, WA 98195-2350 | 206.685.3134
nasmith@cs.washington.edu | https://homes.cs.washington.edu/~nasmith

2017; it was such an exciting piece of work (in my opinion) that I featured it prominently in my keynote talk at the same conference and several talks afterwards.

A follow-on paper at NAACL 2018 expanded the main idea to bridge semantic dependency parsing with FrameNet parsing (adding frame semantics expert and then-student Swabha Swayamdipta, who will join USC's faculty in 2022); that paper was a true team effort by the three students. The main method jointly parses into both representations at the same time using a shared neural-structured predictor. An important idea here was the use of an L_1 regularizer on the (expensive) cross-task factors, pruning many of them away, and leading to model that was actually possible to run; Hao worked out how to do this effectively in implementation. It was in working on this project that I realized, first, how effectively Hao works on a team and, second, how strong his engineering skills are. (The mathematical skills needed to combine neural networks with AD^3 were already in evidence in Hao from the project the year before.)

In the two projects above, Hao's focus was on models that share parameters and can be used for more than one kind of semantic structured output. In the next line of work, he turned to a more traditional framework, in which structures are produced serially: pipelines. In a traditional pipeline, each module is built independently, and the output of one is passed as part of the input to the next, for example, a syntactic parse is generated and fed to a semantic analyzer. Such approaches have attractions (modularity and interpretability) but are prone to error propagation. They are out of favor at present, since the current aesthetic prefers "end to end" training of models without concern for intermediate representations that might be recognizable to humans; intermediate representations are, in this paradigm, to be learned entirely and never need to be explicit. The practical challenge of pipelines when we use deep learning is that we can't backpropagate error signal to parameters through hard (discrete) decisions. Hao developed, with Sam Thomson, the structured projection of intermediate gradients optimization technique (SPIGOT). This was a project where the level of mathematical sophistication required was beyond mine; Hao and Sam deserve all the credit for developing the idea. The final paper is really cool, both theoretically and experimentally; it's one of a very small number of results I've seen where a parser directly contributes to significant improvements on a downstream task (sentiment analysis) over a strong baseline. The work was published at ACL 2018 and recognized with **honorable mention for best paper**.

Over the past couple of years, a new paradigm has come to dominate much of NLP research. A large neural network is "pretrained" on a vast amount of unannotated text, usually to predict some words given others. This network is then used to initialize a very slightly more complex network that is then "finetuned" through supervised learning to carry out a specific task. Much of the linguistics-inspired NLP "pipeline" of decades past – where we break sequences into tokens, add part-of-speech tags to them, parse them into trees, etc. – has been largely abandoned. This is an interesting development, leading to extensive work on "probes" that try to determine the extent to which the pretraining is essentially rediscovering features that encode those linguistic descriptions. Much of this analysis (including by my own group) has been focused on syntax. What about semantics?

Hao teamed up with masters student Zhaofeng Wu (now applying to Ph.D. programs) to explore how the semantic dependency graphs discussed above midhgt be used in the pretrain-finetune paradigm. They applied Hao's strong semantic dependency graph parsers to sentences to be finetuned. The representations from a pretrained transformer model (RoBERTa) were coupled with representations derived from a graph convolutional neural network applied to the semantic graph; these were finetuned together toward a task of interest. They carried out a set of semantic probing experiments to measure the extent to which predicate-argument structures were, in some sense, already discovered, using syntactic dependencies as a point of comparison (bottom line: they are not). The main experiments in the paper showed a small but consistent gain across the GLUE benchmark. Smaller gains (on par with using a syntactic parser) are possible when the semantic graphs are used only at training time (not test time). The paper includes extensive analysis (e.g., diagnostic datasets, learning curves, alternate architecture choices). The work was accepted to the *Transactions of the ACL* (one of our two top journals in NLP).

The end result, "SIFT" (semantically-infused finetuning) is a creative technique that lets us use both pretrained transformers and linguistically-sophisticated parsers to achieve task improvements.

The story of Hao's contributions to structured prediction for NLP is notable for how well he's navigated ever-changing waters, often anticipating major waves uncannily well. If these projects constituted his Ph.D., it would have been enough, but I think his greatest impact to date has actually been elsewhere!

Rational recurrences. Hao teamed up with Sam Thomson and then-postdoc Roy Schwartz (now on the Hebrew University faculty) to develop the construction of "rational recurrences": recurrent neural networks that can be rewritten as neural finite-state machines. When I talk about rational recurrences, I feel obligated to point out that the term *rational* here is inherited from theoretical algebra (e.g., *rational series*); it is not intended to brand the neural networks as somehow displaying rational behavior.

By taking this view of recurrent architectures, many existing models were unified into a family and a new lens could be placed on their differences; LSTMs meanwhile were conjectured (and later shown) not to be rational. A new way to design models with interpretability "baked in" emerged. As a proof of concept, the team introduced a new rational RNN and showed that, with far fewer parameters than an LSTM and far weaker expressive power, did a better job of language modeling and text classification. The resulting paper at EMNLP 2018 was, in my view, extremely exciting and pointed a way forward for neural NLP that brings back some of what was lost when we moved away from earlier models: an understanding of the internals and hence a way to characterize what models do well and where they might go wrong. Hao was the primary implementer and experimenter (though he also contributed significantly to the theoretical work) and presented the paper at EMNLP. In the corridors, I heard many compliments on his presentation and the ideas in it, one quite memorable (a student of a respected colleague telling me, on the verge of joyful tears, that the paper radically changed his thesis research plans).

Rational recurrences launched a range of new projects, ranging from the highly theoretical to the extremely practical. On the theoretical side, Will Merrill (a predoctoral mentee of mine at AI2, now a Ph.D. student at NYU) used Hao's notion of rationality to establish a formal hierarchy of recurrent neural network architectures (somewhat analogous to the Chomsky hierarchy for formal languages), deepening our understanding of how rational recurrences, LSTMs, and the now-dominant transformer networks' expressive capacities relate to each other. On the practical side, Hao himself used rational recurrences to efficiently expand neural language models with hierarchical structure (PaLM: a hybrid parser and language model, EMNLP 2019). With Jesse Dodge (a Ph.D. student in our group), he used them to introduce structured sparsity into neural NLP models. This was an extremely elegant result: by forming parameter groups associated with states in the finite-state network backbone of the recurrent network, and encouraging those groups to go to zero, Hao and Jesse learned compact finite-state machines that were a tenth the size of the original models, but with competitive performance. Though there has been a lot of work on trying to compress or distill large neural networks down to manageable size after the fact, this work stands out for its use of structured sparsity, which I've seen as a way to incorporate inductive bias and expert knowledge of the domain and task, and also to achieve more interpretable models. This work was also published at EMNLP 2019.

Transformers and efficiency. In the past couple of years, however, the mainstream of the field has not been moving toward "compact" or "interpretable" models. The transformer architecture, introduced in 2017 by researchers at Google, was designed for commensurability with GPU hardware, not natural language data or theories of language. It's difficult to overstate the importance of the transformer architecture; since it was introduced it has affected virtually every area of research and practice in NLP (and is also gaining traction in computational biology and computer vision). It has enabled a dizzying "scaling up" of NLP systems, both in terms of model size and training dataset size. A widespread concern in academia is that the best models are now too large for most researchers to work with, requiring massive hardware investments. There are three concerns: this situation is not inclusive (only rich industry labs can train big transformer models), it is costly, and it leaves a massive carbon footprint.

Efficiency has emerged as an important research goal that has inspired the latest stage of Hao's research.

The transformer model is simple enough to explain to undergraduates in one lecture, but it doesn't lend itself well to any kind of formal interpretation; its self-attention mechanism is a very dense nonlinear transformation applied repeatedly. In paper published at ICML 2020 a team in Switzerland reanalyzed transformers in terms of dot-products of feature maps, yielding a linear-time approach to inference and training – hundreds to thousands of times faster than the quadratic algorithms used previously, on a range of tasks. Apart from this huge practical benefit (I really can't understate how broadly useful it is!), the analysis leads to a new way to look at transformers as recurrent neural networks. This finding is fascinating, bringing the supposedly radically-new transformer model back into the fold of models to which a growing body of theoretical results applies (including Hao's work above).

Building on these ideas, Hao teamed up with internship mentors at Deepmind and Nikolaos Pappas, then a postdoc in my lab and one of the authors of the ICML 2020 paper above, to develop **random feature attention**, which combines the kernel view with a random feature approximation for further speedups, allowing application to longer sequences at low cost. The experiments showed exciting benefits to machine translation and language modeling. This work was published at ICLR 2021 and was a spotlight paper.

A follow-on project, led by PhD student Jungo Kasai and Hao and including collaborators at Microsoft, Deepmind, and UW, showed how to use the ideas in the random feature attention paper to transform a transformer language model into a recurrent neural network. This might seem like a step backwards, but RNNs have a special computational advantage over transformers when *generating* text autoregressively: they are far more efficient. Specifically, they propose a swap-then-finetune procedure: in an off-the-shelf pretrained transformer, replace the softmax attention with its linear-complexity recurrent alternative and then finetune. With a learned feature map, the approach provides an improved tradeoff between efficiency and accuracy over the standard transformer and other recurrent variants. They also show that the finetuning process has lower training cost relative to training these recurrent variants from scratch. As many models for natural language tasks are increasingly dependent on large-scale pretrained transformers, this work presents a viable approach to improving inference efficiency without repeating the expensive pretraining process. The work was published at EMNLP 2021.

Hao's latest project, in typical Hao fashion, builds on his past insights but brings a totally new perspective to the object of study. The core of the transformer architecture, self-attention, can be understood as a random-access memory, with each word token in the context of a particular calculation occupying one slot. Under this view, memory size must grow linearly with the length of an input sequence, as well as the overhead of reading from it. A few researchers have proposed to bound the memory size in various ways; Hao unifies them into a single abstraction, **attention with bounded-memory control** (ABC). Different approaches organize the memory differently, and the ABC abstraction opens up new, unexplored possibilities. For example, it shows how to incorporate the much-discussed Linformer approach in a left-to-right model. Hao introduces a new approach to organizing the memory using a learned, contextualized strategy rather than the heuristics of past approaches. Experiments on language modeling and machine translation show that it outperforms previous efficient attention models and improves significantly the inference time and space efficiency of the strong transformer baseline with no or negligible accuracy loss. The work is under review.

In the midst of this work on efficiency, Hao contributed significantly to the vision and writing of an **NSF grant** (UW portion $500K, funds partially matched by the BSF to support work at Hebrew University), which was awarded on the first attempt. Early on, he was awarded a prestigious **Google fellowship** – his work is eminently fundable and he's already adept at writing compelling proposals.

Other projects. A strong sign that a Ph.D. student will grow into a leader is the breadth of projects they take on that are tangential to their primary pursuits. Not all students manage to do this, but those who are especially adept at time management and who cultivate a collaborative attitude can achieve an impressive range. Hao is one of those; some examples:

- Early on, Hao collaborated with Chenhao Tan (formerly a postdoc with us, now assistant professor at the University of Colorado at Boulder) and me on a **computational social science** project. This work was quite different from Hao's other interests (I encourage exploration in the early years of the Ph.D.): the question we asked was about the choices that news outlets make when selecting content from presidential debates to highlight in reports over the days following the debate. Are these choices predictable? What linguistic features are predictive? The study involved a significant data collection effort, experiments with human judges (who, it turns out, are not very good at guessing what parts of a debate will be quoted), and the design of linguistic classifiers. The project was largely Chenhao's idea, but Hao was a great contributor who accelerated the pace of progress and had important insights at critical times during the project. The work was published at WWW 2018. I mention it because it's rare for an "algorithms-first" student like Hao to engage so deeply with a "data analysis" project of this kind.
- One of the longstanding application areas of NLP is translation between languages (known as "MT"). The aforementioned transformer architecture was originally introduced as a model for MT (by Ashish Vaswani and colleagues at Google). Together with Jungo Kasai, Facebook's James Cross, and Nikos Pappas, Hao considered the tradeoffs inherent in a major design decision in MT modeling. The more traditional route is to build an MT model that is like a language model: it predicts words one at a time, left to right, in what is often called an "autoregressive" fashion. A more recent alternative is to decode all words in parallel, without directly modeling the interactions of the words. This "non-autoregressive" MT is very fast, and works better than I would have expected, but performance lags behind autoregressive models, in terms of translation accuracy on held-out data. The group recognized that, in the (slower) autoregressive model, some of the computation can be easily parallelized (the "encoding" part of the model); it is only the decoding layers that must work left-to-right. In practice, they find that making the encoder "deep" (more neural network layers) does not slow down translation, and making the decoder "shallow" does not harm accuracy. Their deep-encoder, shallow-encoder approach gives a much better speed-accuracy tradeoff, especially when translating in large batches, than earlier autoregressive approaches as well as the non-autoregressive approach. This work was published at ICLR 2021, and it was in the top 3% of submissions ranked by average reviewer score.
- Another project considers yet another dimension of the transformer architecture (above, we looked across its layers, across the dimensions of its hidden representations, across its memory). Now we consider the parallel "heads," copies of the same attention module at each layer that learn and predict in parallel. Hao draws a parallel between these heads and mixture of experts models, proposing to reallocate heads so that they specialize on different inputs. Using a novel training algorithm based on block coordinate descent to encourage this behavior, Hao achieved predictive gains on language modeling and machine translation tasks, over strong baselines. The work was published at ACL 2020.
- In a completely different kind of project, Hao worked with Tongshuang "Sherry" Wu (another UW Ph.D. student and fantastic faculty market candidate), Alexis Ross (a predoctoral scholar on my team at AI2, now applying to Ph.D. programs), and others at AI2 on a project in the area of controlled text generation. Recently the idea has emerged to automatically generate text with specific properties, specifically to synthesize additional training data for NLP models. The team introduced Tailor, which offers controls that are derived from semantic roles (a theory related to the semantic formalisms discussed earlier). They implement a set of operations on control codes that can be composed into complex perturbation strategies, and demonstrate their effectiveness in

constructing challenging “contrast” test sets that are lexically diverse, improving model generalization through data perturbation, and improving compositionality in fine-grained style transfer tasks. The work is currently under review.

In the interest of space, I’ll note that the above discussion is not exhaustive; Hao has made various other research contributions, especially during his internships, to areas as disparate as text generation and adversarial NLP. His breadth is extremely unusual for a Ph.D. student!

Communication. Hao is an outstanding communicator. Early drafts of his papers are always carefully written with utmost attention to detail; they’re a pleasure to help with. His talks are crystal-clear and easy to follow, with gorgeous visuals. Though the UW CSE Ph.D. program – like most others – does not offer a lot of opportunities to learn how to teach well and I expect all of our graduates will, like most of us, have to spend some of their early-faculty time figuring out how to teach, Hao’s natural communication abilities will put him ahead of his peers in this regard, and will be a boon to his mentees.

Mentoring. Hao is generous in advising other researchers, even more senior ones, not just on their papers and presentations, but on their larger visions. Though Sam Thomson (about four years senior to Hao) brought a powerful and creative intellect to the Ph.D., collaborating with Hao always gave him an endurance boost, and I think that’s a big part of how he got past the finish line. Hao’s junior colleagues, Jungo Kasai, Sofia Serrano, Ofir Press, and Alisa Liu, view him as a kind of archetype of (one kind of) extremely successful Ph.D. student, but not just an example to follow, also a friendly, approachable person who can be relied upon to give great advice.

He has also spent considerable time working with undergraduate and masters student researchers. I mentioned Zhaofeng Wu above; there are four others. Another, Michael Zhang, is now a Ph.D. student at the University of Texas, and three more are actively working with Hao now on a range of different projects. I explicitly encourage risk-taking and creativity – not publication – in the undergraduate projects that my Ph.D. students and postdocs design and mentor, and Hao has done a great job of helping to train newcomers to the field to think deeply about their choices of problems and how their experimental work will form an illuminating argument. Watching these students grow (I meet with them as a group weekly, leaving detailed day-to-day project-specific mentoring to the Ph.D. students and postdocs), I wish I could go back and do a Ph.D. with Hao.

Service and leadership. Hao serves as a reviewer for all of the top NLP venues (ACL, NAACL, EMNLP, and the TACL journal) as well as the top machine learning venues (ICLR, NeurIPS, and ICML). I’ve seen his reviews from the chair/editor side and they are constructive, thorough, and scholarly.

It would be offensive to call Hao a “machine” – each paper he writes contains an exciting story that ties together foundational questions and well-crafted experimental investigations. As he has matured, he’s continually raised the bar and held his work to ever-higher standards (though his bar was very high from the start!), showing intense awareness of the developments of the field and his place within them.

I’ve been fortunate to mentor a number of star researchers focused on core problems in NLP. Though what is “core” to NLP has shifted dramatically even in the past five years, Hao’s contributions and interests really are at the heart of the field, building simultaneously on insights from linguistics, machine learning, and computation without skewing heavily toward any of these foundational viewpoints, as NLP has goals of its own. In this regard, he’s probably most like my earlier Ph.D. students Andre Martins (faculty at the University of Lisbon and head scientist at Unbabel) and Dani Yogatama (Deepmind) and certainly matches their brilliant abilities at their graduation times, as well as their breadth of interests. All three of them have the uncanny ability to look mathematically at NLP’s ubiquitously used tools of the moment and see something new, and then execute projects that *change the way we think* about those tools and the problems they solve. Though Hao is among the most mathematically sophisticated students I have mentored and learned from, he learned faster and with less effort how to translate his ideas to others with less sophistication; for this reason, I believe he's especially well poised to build a lab of diversely talented students and far-ranging interests.

In terms of recent NLP hires in academia, there have been relatively few focused at this core; more frequently we see departments hiring experts focused on application areas (e.g., question answering or machine translation) or interdisciplinary junctions with computer vision, robotics, or the social sciences. Hao is the ideal hire for a department seeking to develop exciting new connections between NLP and the full spectrum of machine learning, from theory to systems-infused AI, though he will also be a fantastic resource for applications-focused or interdisciplinary researchers. Given his foundational focus, I can't imagine anyone in academia (at any level, myself included) will be able to teach a better NLP course than Hao.

Summary: To close, Hao is the kind of researcher whose thoughts everyone wants to hear at the end of an invited talk: he brings a unique perspective that seasons the conversation and gets everyone thinking. He is ambitious, creative, and rigorous. He brings with him a huge network of talented collaborators around the world (all the top industry research labs and some of the most visible academic groups in Asia, Europe, and the US are named on his CV). He is an intellectual force to be reckoned with, who consistently produces technical research that is highly visible and highly impactful. **I recommend him to you as a faculty colleague you'll be delighted to meet, learn from, and work with, in the strongest possible terms, and with no qualifications.**

If there is more information that I can provide, please contact me at nasmith@cs.washington.edu.

Best regards,

[illegible]

Noah A. Smith
Amazon Professor of Machine Learning
Paul G. Allen School of Computer Science & Engineering
University of Washington

Senior Research Manager
Allen Institute for AI

(You can read a bit more about me at https://nasmith.github.io/.)

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

December 1, 2021

Re: Letter of recommendation for Hao Peng

Dear Hiring Committee,

I am writing to provide very strong support for Hao Peng's application for your open faculty position. I have never had the opportunity to work with Hao directly, but I have closely followed his work during his PhD, as it is highly innovative and in areas of personal interest. Hao has an incredibly track record of doing technically sophisticated work that deepens our understanding of existing workhorse natural language process (NLP) models, while also introducing new methods to make them more efficient or otherwise improve their learning and generalization. He has a number of results that I find surprising and has shed new, formal insights on long standing confusions for the field, as I will summarize in more detail below. This work is particularly impressive in the modern era of deep learning for NLP, where the field as a whole has become incredibly empirical, with relatively little effort to understand how or why our methods work. Hao's research provides, over and over again, shining examples of how we can aim for more, by designing algorithms with better formal guarantees and introducing new ways to understand how our newer methods relate to the tens of years of work that came before. This style of work is sorely needed, and one of the many reasons I am very happy to recommend Hao without reservations. He is an ideal candidate that I hope you will consider very closely.

Hao's most recent research thrust has been on efficient methods for natural language processing. This work falls loosely in the area of Green AI, which has formed recently in response to the growing computational costs of building state-of-the-art models (e.g. language model pretrianing can take 100s of thousands of GPU days for a single model). Hao has made significant contributions to nearly all of the key bottlenecks / open challenges in this area. Modern models include so-called self attention layers that have O(n^2) space and time complexity for an input string of length n. Essentially, every token compares its associated vector to the vectors for all other tokens, allowing for very fast global exchange of information (and much better performance!) at a very high cost (but still possible due to the

Luke Zettlemoyer
Professor
Box 352350 / Seattle, WA 98195-2350 / 206-685-1227
lsz@cs.washington.edu / https://www.cs.washington.edu/people/faculty/lsz/

incredible parallelism that comes with GPUs). Although many different efforts have been made to reduce this complexity, they were all ad hoc, with no general theory of what needs to be computed to more efficiently get the required global information exchange. Hao introduced a new approach that unified and generalized many of these methods, based on ideas from hashing random feature projections, thereby achieving significant speed gains without sacrificing accuracy. This was a strong achievement that surprised me and, despite being new, is quickly becoming the standard approach in this area. Hao has also looked at a range of other efficiency challenges with similarly impressive gains, including showing how to get efficient self-attention into language model fine tuning schemes and how to speed up machine translation systems by better balancing the compute assigned to their encoder and decoders. Overall, this is a new direction for Hao, but one that is already bearing fruit and will provide a strong foundation for many years of exciting work to come.

Earlier in his PhD, Hao also had a number of very strong advances on methods that build structured semantic representations of an input sentence, for example including semantic meaning graphs. This work really stood out in his ability to combine modern neural methods (which were just emerging at the time!) with models and objectives that explicitly represent and reason about output structures (following much work that was popular before deep learning took off!). How had too many strong results in this area to list them all. However, highlights include (1) new methods for using structural inductive biases to improve the loss function used to train semantic parsers, (2) algorithms for end-to-end training of NLP pipelines which cleverly relax the inherently non-differential nature of these problems, and (3) new techniques for learning from complementary signals provided by multiple meaning representations, to enable fundamentally new types of semi-supervised learning for low resource representations (that don't have much training data). This work strongly established Hao as a mathematically sophisticated researcher, in every case including carefully designed and relatively formally well understood methods, that went well beyond what most of the field could develop. This work has also had significant impact, including winning a paper award nomination at ACL (the top NLP conference), inspiring many follow up papers, and even convincing groups of researchers to form shared tasks to compete on improving some of the results.

Finally, Hao is perhaps best known for his work on formal analysis of deep learning methods, which is also my personal favorite of his lines of work (although perhaps also farthest from my area of expertise). Here, Hao has been establishing a number of results that relate neural networks to other models of computation (e.g. finite state machines) that have previously been much more carefully and

Luke Zettlemoyer
Professor
Box 352350 / Seattle, WA 98195-2350 / 206-685-1227
lsz@cs.washington.edu / https://www.cs.washington.edu/people/faculty/lsz/

formally studied by the NLP research community. His work on rational recurrences showed that single layer workhorse neural networks (e.g. RNNs and CNNs) are actually equivalent to a particular type of weighted finite state machine. He also went beyond just this theoretical analysis to show that learning rational recurrences directly leads to more efficient and interpretable neural models, a surprising result that shows the generality of the relationship. Overall, the field desperately needs a deeper understanding of how our methods relate to each other, especially ones like this that also drive new empirical advances. Hao is one of the very few researchers that are working in this field, and I have no doubt he will provide a shining example that many others will follow.

Although I have never seen Hao teach, I have seen him give very high quality research talks, and have no doubt these skills will transfer to classroom settings. He has also been an excellent mentor to a number of junior students and has a particularly strong external profile, giving a large number of invited talks in the last two years and winning multiple awards, including a Google PhD fellowship and an ACL paper award (the top NLP conference). Hao also has a compelling research vision, as outlined in his research statement, and is a veritable fountain of new ideas. Overall, Hao is the complete package, as ready as you can be to start the transition to faculty life!

In summary, Hao is a creative researcher with both significant technical depth and breadth that strives to introduce new formally well understood methods, while also drawing new connections to previous generations of methods to improve the foundation for modern work. He has also introduced some of the most efficient and scalable methods in NLP to date, again building on the insights from his more formal understanding of the field. He is already a clear thought leader in these spaces, easily the best on the market this year, and I have no doubt he will only accelerate even more as he moves to faculty life. For all these reasons, it is a pleasure to recommend Hao without reservation, and I hope you will consider him very carefully for your open faculty position. I think he will be one of the top NLP candidates this year, and have no doubt he will be very successful at any top department that manages to hire him. Please do not hesitate to reach out if I can help in any way.

Sincerely,

[illegible]

Luke Zettlemoyer

Luke Zettlemoyer
Professor
Box 352350 / Seattle, WA 98195-2350 / 206-685-1227
lsz@cs.washington.edu / https://www.cs.washington.edu/people/faculty/lsz/

Bio: Luke Zettlemoyer is a Professor in the Paul G. Allen School of Computer Science and Engineering, and a Research Scientist at Meta. His research focuses on empirical methods for natural language semantics, and involves designing machine learning algorithms, introducing new tasks and datasets, and, most recently, studying how to best develop self-supervision signals for pre-training. Honors include multiple paper awards at the top NLP and ML venues, a PECASE award, an Allen Distinguished Investigator Award, and being named to the DARPA Computer Science Study Group. Luke received his PhD from MIT and was a postdoc at the University of Edinburgh.

Luke Zettlemoyer
Professor
Box 352350 / Seattle, WA 98195-2350 / 206-685-1227
lsz@cs.washington.edu / https://www.cs.washington.edu/people/faculty/lsz/

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Yejin Choi
Professor
Paul G. Allen School of CSE
University of Washington
&
Senior Research Manager
Allen Institute for Artificial Intelligence
http://homes.cs.washington.edu/~yejin

Dec 1, 2021

Dear the Faculty Search Committee,

I am writing this letter to provide my most strong endorsement for **Hao Peng** for a tenure-track faculty position at your institution. While I have never collaborated or even directly interacted with Hao, I know his work reasonably well as Hao is a student of my dear colleague Prof. Noah Smith at the University of Washington.

Let me cut to the chase and state the obvious: Hao is a super star. He is an absolutely impressive and exceptional candidate at the cross between Natural Language Processing (NLP), Computational Linguistics, and Deep Learning. After just 5 years into his Ph.D. study, Hao already published 19 papers at top venues such as ACL, EMNLP, TACL, NAACL, and ICLR, and 11 of which as the lead author. While the number of papers at top venues is not a meaningful measure of the originality or the impact of one's research, this level of productivity is still undeniably exceptional and noteworthy. It is probably reasonable to assume that Hao is the best in N year type student from Prof. Noah Smith's group.

The level of Hao's productivity is even more impressive when we consider the intellectual significance and impact of Hao's research. The impact of his work has been recognized through the Google Ph.D. Fellowship, the Honorable Mention for ACL 2018, the Heidi Hopper Endowed Regental Fellowship, and several additional fellowships from Peking University. Even the number of fellowships Hao earned might be the largest I have seen. In addition, Hao has been invited to give talks at prestigious places such as DeepMind, the University of Alberta, the University of Hongkong, the University of Pecking, and New York University at Shanghai, demonstrating that Hao has already established a strongly visible identity in the research community through his work.

Hao's research aims to address several important challenges of NLP: sample-efficiency, robustness, generalizability, and interpretability. Toward these goals, Hao designs algorithms, new learning paradigms, and formal analytic methods. The three most significant threads of research Hao has achieved through his Ph.D are: (1) efficient deep learning methods for NLP, (2) deep learning with structural inductive biases and (3) formal analysis of deep learning methods for NLP, as further elaborated below.

Efficient deep learning methods for NLP For efficient deep learning methods, Hao has developed attention mechanisms with linear complexity to significantly improve the efficiency of the transformer architecture. The conventional attention mechanism of transformers operates with a quadratic complexity in the input length, which has been the major computational bottleneck when scaling up the context size. Thus there have been recent studies that aim to "compress" the context via various strategies. The intellectual contribution that Hao's work brings to the field is a unified abstraction across these existing approaches, which then also inspires a new elegant approach that learns to compress the context dynamically. Hao's final approach achieves positive empirical gains over prior approaches.

Hao then goes one step further to propose Random Feature Attention (RFA) that draws inspiration from traditional kernel methods in order to derive linear-complexity approximation to the canonical attention mechanisms. This is a highly creative work that also achieves impressive empirical performance, as RFA can achieve significant time and memory saving without any accuracy loss. This paper was a **spotlight at ICLR 2021**.

Deep learning models for NLP with structural inductive biases Another line of research that Hao has been pursuing is making NLP models more structured and compositional, to reflect the correct linguistic insights and biases into the monolithic neural networks. With the advent of structureless neural networks, it has becoming almost as if structural linguistic representations are empirically harmful against robust representation learning. Or, that's what a lot of researchers in the research community have assumed for a while. Hao's research investigates this more carefully to identify exactly where the fundamental problems are, and proposes an elegant solution to learn to benefit from the linguistic structures successfully. In particular, Hao's work demonstrates that it is possible to draw benefits from the symbiosis among multiple semantic representations, even when they are disjoint with no parallel annotations, and even when they are from structurally divergent resources.

In another work, Hao has demonstrated an end-to-end learning approach called SPIGOT, which successfully incorporates structure-based NLP pipelines to yield more accurate and interpretable decisions in downstream tasks. The significance of SPIGOT has been recognized by **an honorable mention for the best paper award at ACL 2018**.

Formal principles of deep learning models for NLP The last line of research that Hao has been leading is to develop formal principles to understand modern neural architectures for NLP, which in turn, can lead to more efficient, interpretable, and effective NLP models. Hao studies the connections between modern recurrent neural network architectures and weighted finite-state automata (WFSAs). The theoretical findings prove that a family of recurrent and convolutional networks, at least in their single-layer cases, are indeed WFSAs parameterized with neural networks, hence dubbed with *rational recurrences*. Hao's research demonstrates that rational recurrences are more interpretable and more parameter-efficient, addressing the recurring fundamental challenges of neural networks in the current time that are too compute heavy and too opaque to understand.

In sum,

Hao is a truly exceptional scholar with unique technical perspectives, depth, and originality to tackle some of the most important and hardest challenges of deep learning models in NLP. Thus, I provide my most enthusiastic recommendation for Hao for a tenure-track faculty position at your institution.

Please feel free to contact me at yejin@cs.washington.edu, should you have any question.

Yejin Choi[1]
Professor
Computer Science & Engineering
University of Washington
`http://homes.cs.washington.edu/~yejin`

[illegible]

[1]*About myself:* I am Brett Helsel Professor of the Paul G. Allen School of Computer Science & Engineering at the University of Washington (UW). I am also a Senior Research Manager at the Allen Institute for Artificial Intelligence (AI2), a non-profit research institution that seeks to contribute to humanity through high-impact AI research, where I oversee the project Mosaic focusing on modeling commonsense knowledge and reasoning. I am among the co-recipients of the ACL Test of Time award in 2021, the CVPR Longuet Higgins Prize (test of time award) in 2021, a NeurIPS Outstanding Paper Award, the AAAI Outstanding Paper Award (best paper award) in 2020, the Borg Early Career Award in 2018, IEEE's AI Top 10 to Watch in 2015, and the Marr Prize at ICCV 2013. I received my Ph.D. in Computer Science at Cornell University, and I was an assistant professor at Stony Brook University during 2010 – 2014 prior to my move to UW.

About my research group: My Ph.D. and postdoctoral advisees have become (or are soon to join) faculty at several universities around the world, including **Carnegie Mellon University** (tenure-track in LTI), **École Polytechnique Fédérale de Lausanne (EPFL)** in Switzerland (tenure-track in CS), **Heriot-Watt University** in UK (tenure-track in CS), **Stony Brook University** (non-tenure-track in CS), the **University of British Columbia** (tenure-track in CS), the **University of Cape Town** (tenure-track in CS) in South Africa, the **University of Southern California** (tenure-track in CS), the **University of Maryland** (tenure-track in CS), and the **University of Texas Austin** (tenure-track in CS).