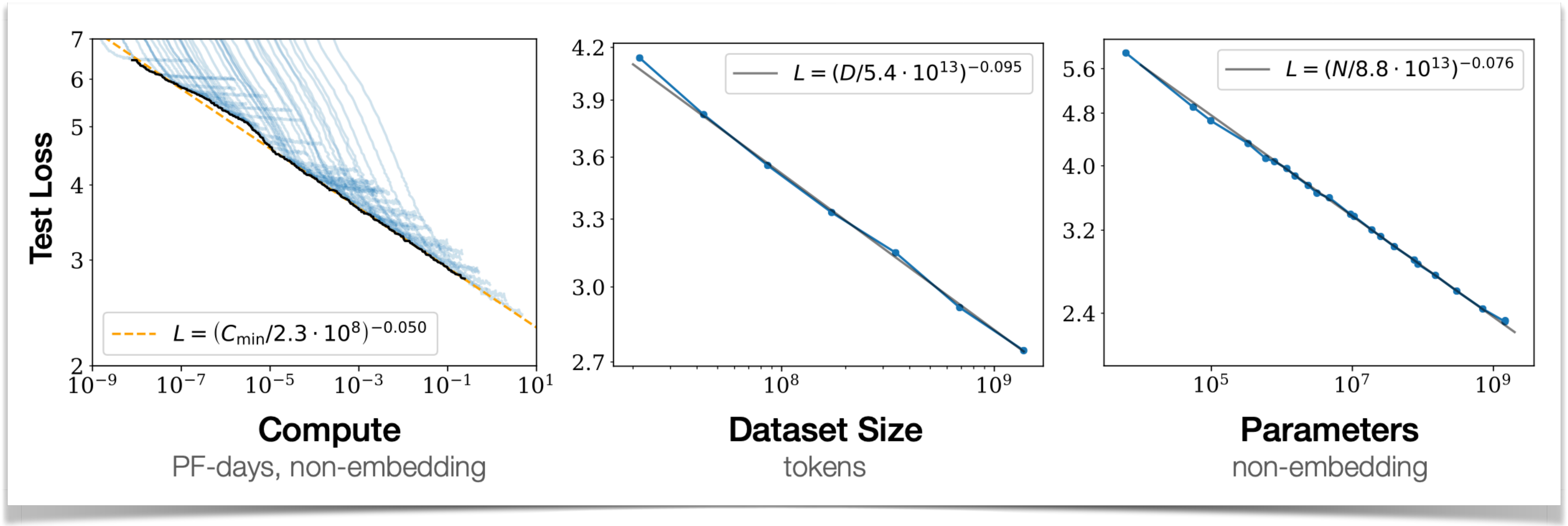
LLMs Are Zero-Shot Problem Solvers — Just Like Modern Computers

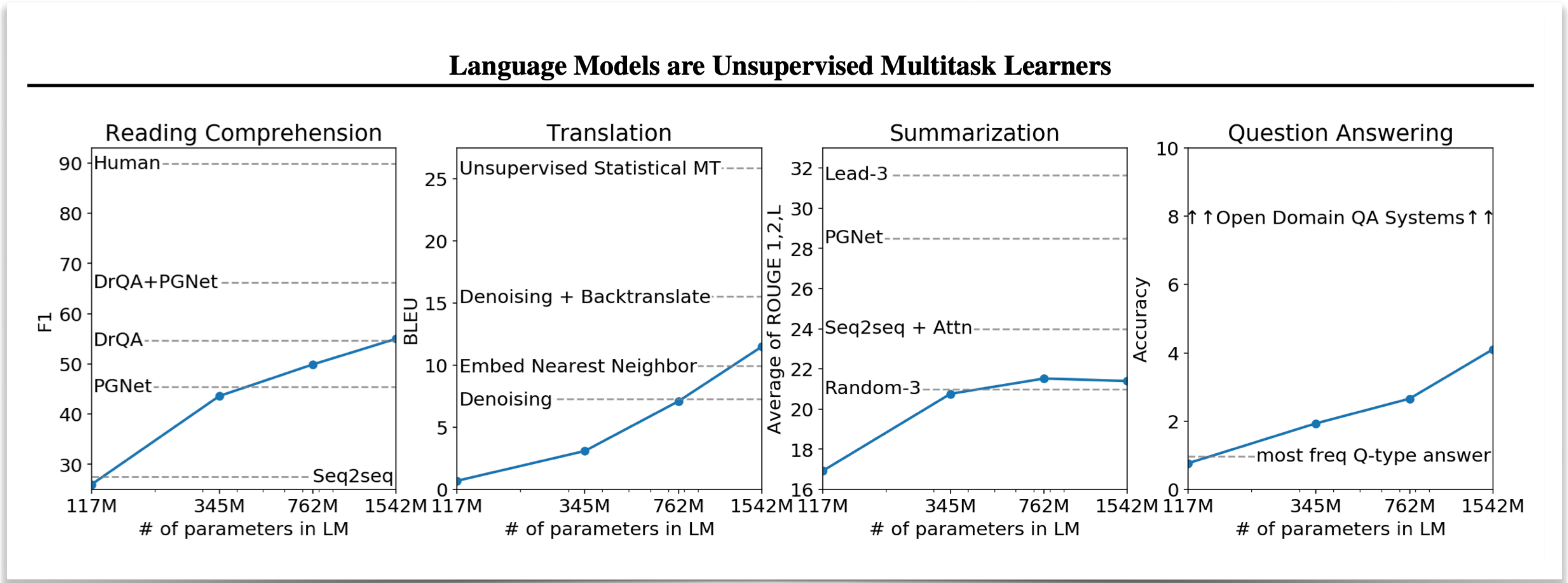
Tim Xiao — FridayTalks

What's so special about LLMs?



- Scaling Laws?
- Not surprising

What’s so special about LLMs?

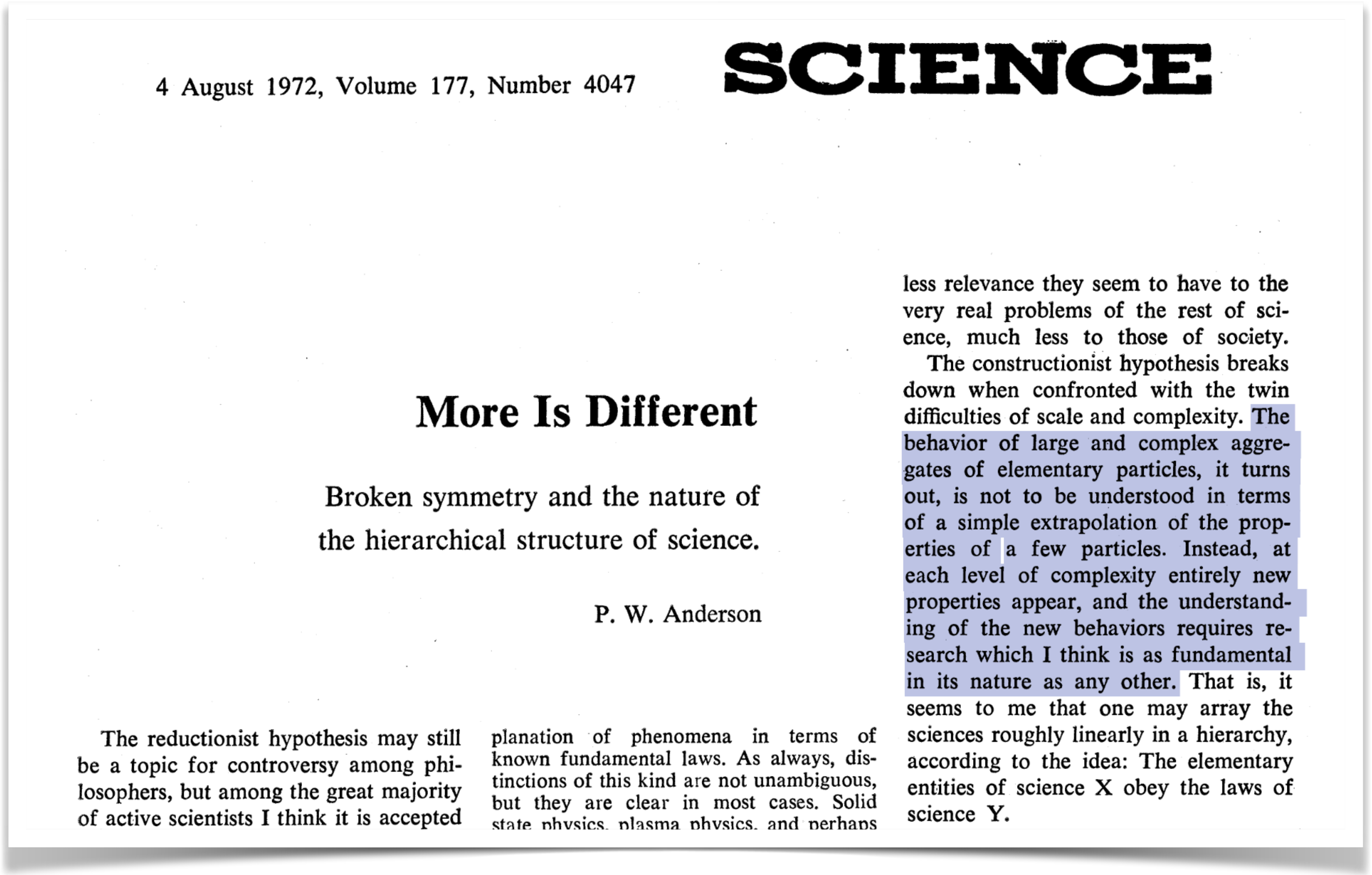


- Trained on LM task, zero-shot other tasks
- Not predicted from the smaller scale models
- Emergence? (Not that special)

Language models are unsupervised multitask learners, OpenAI blog. 2019.

Emergence is not a unique phenomenon!

- Discussed in physics & biology
- "More Is Different"



4 August 1972, Volume 177, Number 4047

SCIENCE

More Is Different

Broken symmetry and the nature of
the hierarchical structure of science.

P. W. Anderson

The reductionist hypothesis may still be a topic for controversy among philosophers, but among the great majority of active scientists I think it is accepted

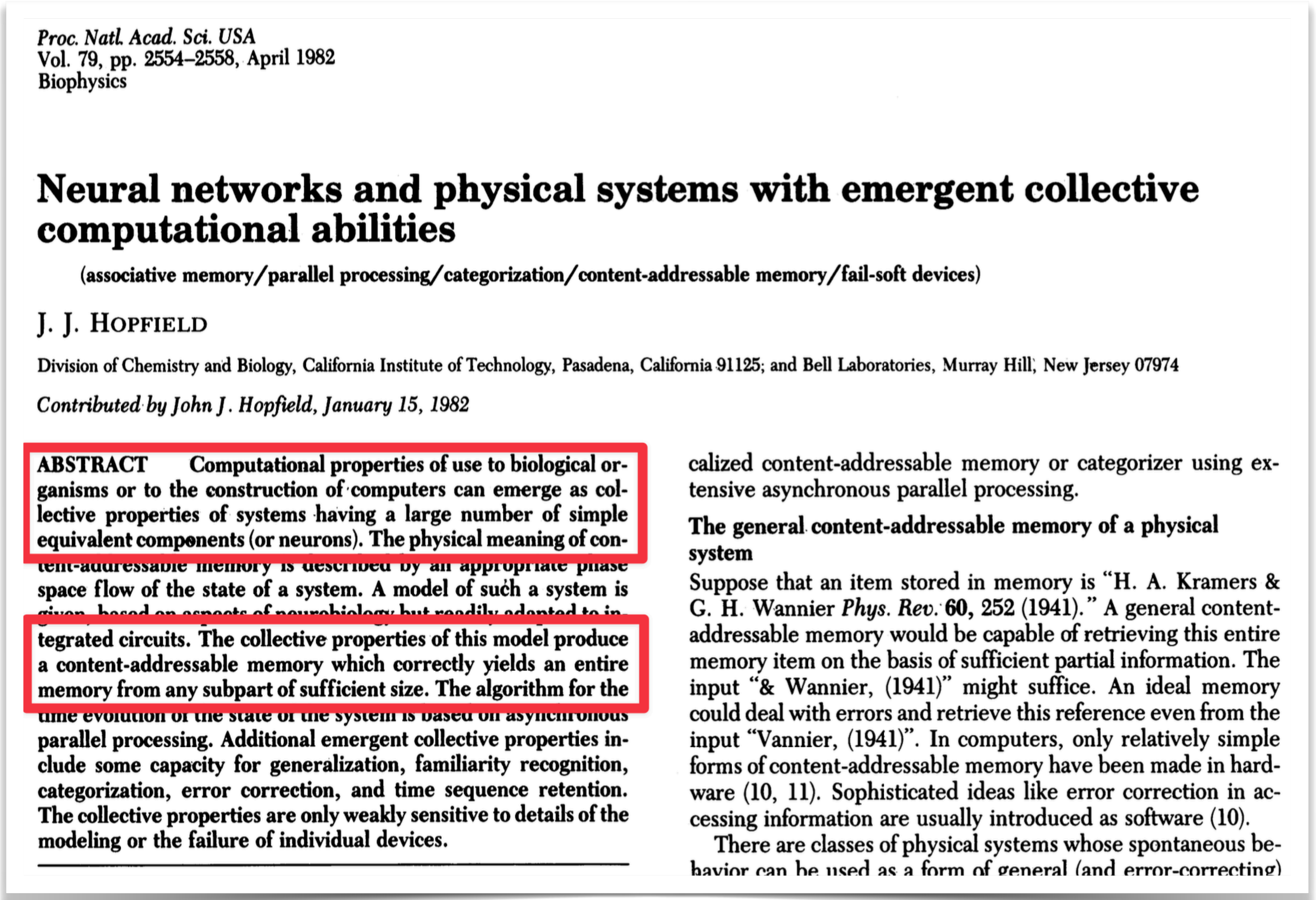
planation of phenomena in terms of known fundamental laws. As always, distinctions of this kind are not unambiguous, but they are clear in most cases. Solid state physics, plasma physics, and perhaps

less relevance they seem to have to the very real problems of the rest of science, much less to those of society.

The constructionist hypothesis breaks down when confronted with the twin difficulties of scale and complexity. The behavior of large and complex aggregates of elementary particles, it turns out, is not to be understood in terms of a simple extrapolation of the properties of a few particles. Instead, at each level of complexity entirely new properties appear, and the understanding of the new behaviors requires research which I think is as fundamental in its nature as any other. That is, it seems to me that one may array the sciences roughly linearly in a hierarchy, according to the idea: The elementary entities of science X obey the laws of science Y.

Emergence is not a unique phenomenon!

- Also not new in machine learning!
- Neuron -> Network
- Hopfield Networks



Proc. Natl. Acad. Sci. USA
Vol. 79, pp. 2554–2558, April 1982
Biophysics

Neural networks and physical systems with emergent collective computational abilities

(associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices)

J. J. HOPFIELD

Division of Chemistry and Biology, California Institute of Technology, Pasadena, California 91125; and Bell Laboratories, Murray Hill, New Jersey 07974

Contributed by John J. Hopfield, January 15, 1982

ABSTRACT Computational properties of use to biological organisms or to the construction of computers can emerge as collective properties of systems having a large number of simple equivalent components (or neurons). The physical meaning of content-addressable memory is described by an appropriate phase space flow of the state of a system. A model of such a system is given, based on aspects of neurobiology but readily adapted to integrated circuits. The collective properties of this model produce a content-addressable memory which correctly yields an entire memory from any subpart of sufficient size. The algorithm for the time evolution of the state of the system is based on asynchronous parallel processing. Additional emergent collective properties include some capacity for generalization, familiarity recognition, categorization, error correction, and time sequence retention. The collective properties are only weakly sensitive to details of the modeling or the failure of individual devices.

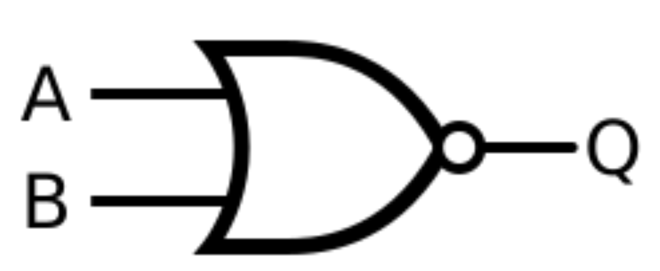
calized content-addressable memory or categorizer using extensive asynchronous parallel processing.

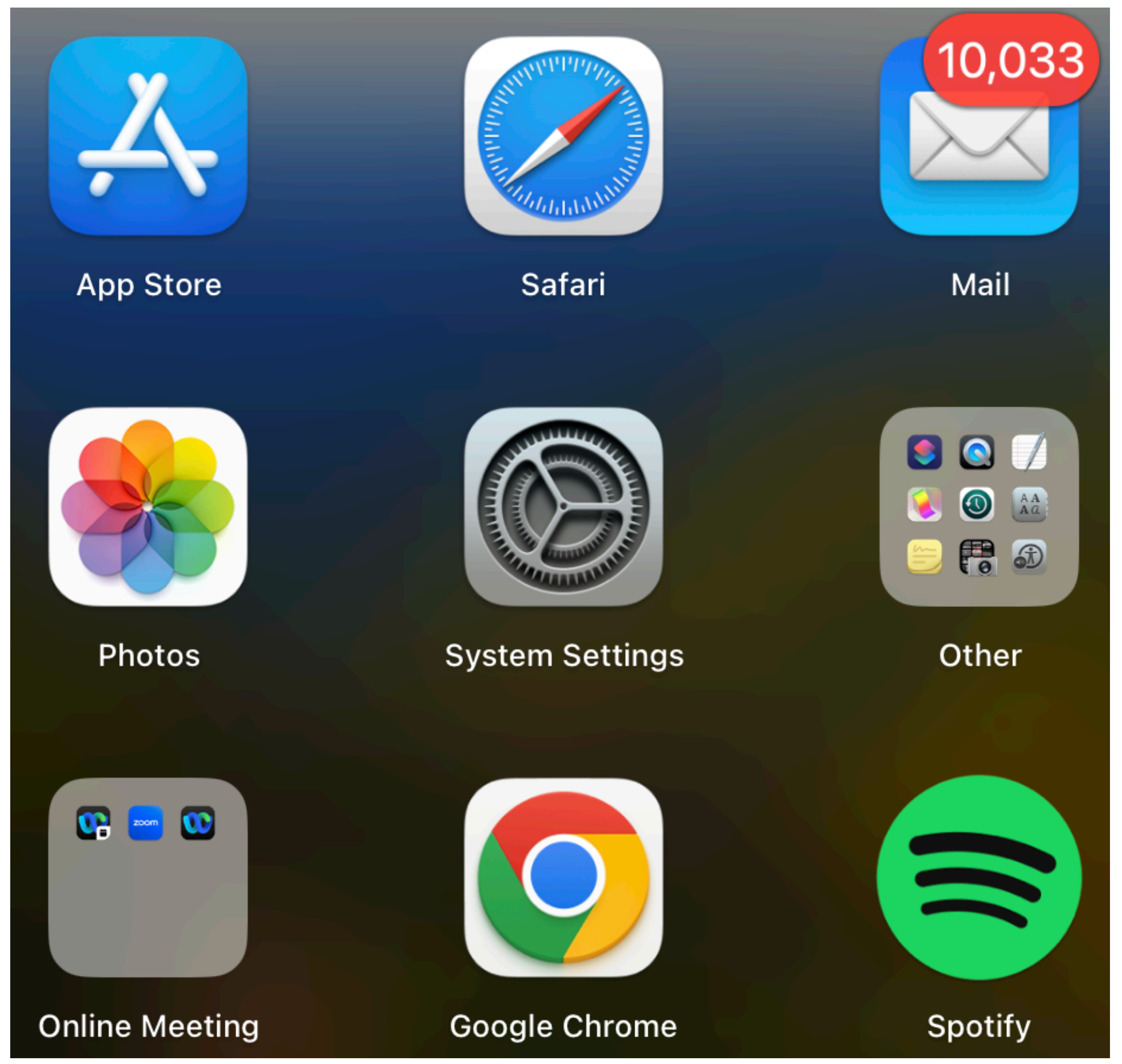
The general content-addressable memory of a physical system

Suppose that an item stored in memory is "H. A. Kramers & G. H. Wannier *Phys. Rev.* **60**, 252 (1941)." A general content-addressable memory would be capable of retrieving this entire memory item on the basis of sufficient partial information. The input "& Wannier, (1941)" might suffice. An ideal memory could deal with errors and retrieve this reference even from the input "Vannier, (1941)". In computers, only relatively simple forms of content-addressable memory have been made in hardware (10, 11). Sophisticated ideas like error correction in accessing information are usually introduced as software (10).

There are classes of physical systems whose spontaneous behavior can be used as a form of general (and error-correcting)

Actually, Modern Computer As Well



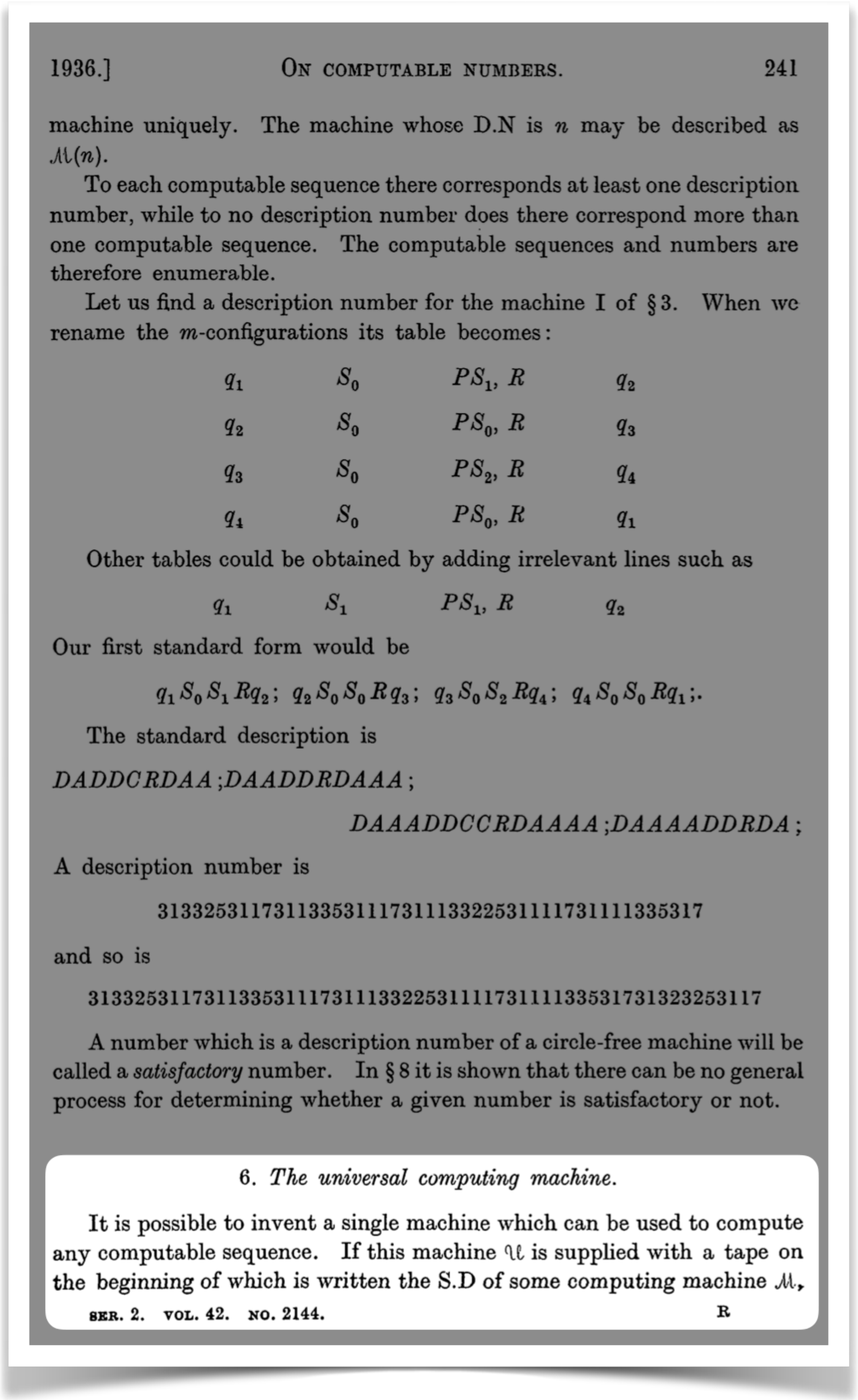


- Also zero-shot solver
- Not built for any specific program but as general purpose machines

A Tale of Two Zero-Shot Problem Solvers

Modern Computers (1936)

- Idea traced back to Alan Turing
- **Universal Turing Machine**
  (i.e., zero-shot)



1936.] ON COMPUTABLE NUMBERS. 241

machine uniquely. The machine whose D.N is n may be described as $\mathcal{M}(n)$.

To each computable sequence there corresponds at least one description number, while to no description number does there correspond more than one computable sequence. The computable sequences and numbers are therefore enumerable.

Let us find a description number for the machine I of § 3. When we rename the m-configurations its table becomes:

q_1	S_0	PS_1, R	q_2
q_2	S_0	PS_0, R	q_3
q_3	S_0	PS_2, R	q_4
q_4	S_0	PS_0, R	q_1

Other tables could be obtained by adding irrelevant lines such as

q_1	S_1	PS_1, R	q_2

Our first standard form would be

$$q_1 S_0 S_1 R q_2;\ q_2 S_0 S_0 R q_3;\ q_3 S_0 S_2 R q_4;\ q_4 S_0 S_0 R q_1;.$$

The standard description is

DADDCRDAA;*DAADDRDAAA*;

DAAADDCCRDAAAA;*DAAAADDRDA*;

A description number is

31332531173113353111731113322531111731111335317

and so is

3133253117311335311173111332253111173111133531731323253117

A number which is a description number of a circle-free machine will be called a *satisfactory* number. In § 8 it is shown that there can be no general process for determining whether a given number is satisfactory or not.

6. *The universal computing machine.*

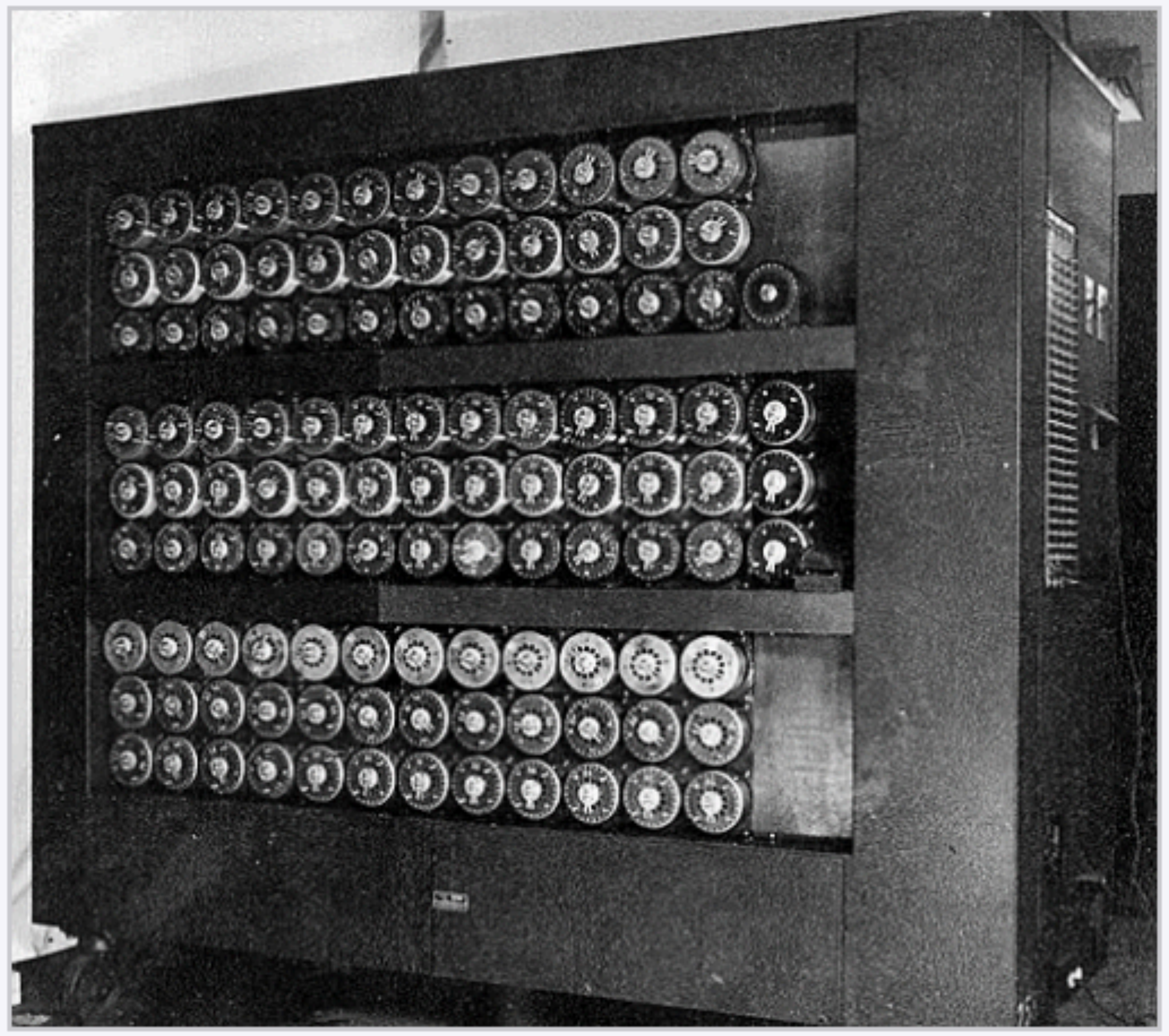
It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $\mathcal{U}$ is supplied with a tape on the beginning of which is written the S.D of some computing machine $\mathcal{M}$,

SER. 2. VOL. 42. NO. 2144. R

A Tale of Two Zero-Shot Problem Solvers

Modern Computers (1939)

- **Special Purpose Computer**
- One machine for one task

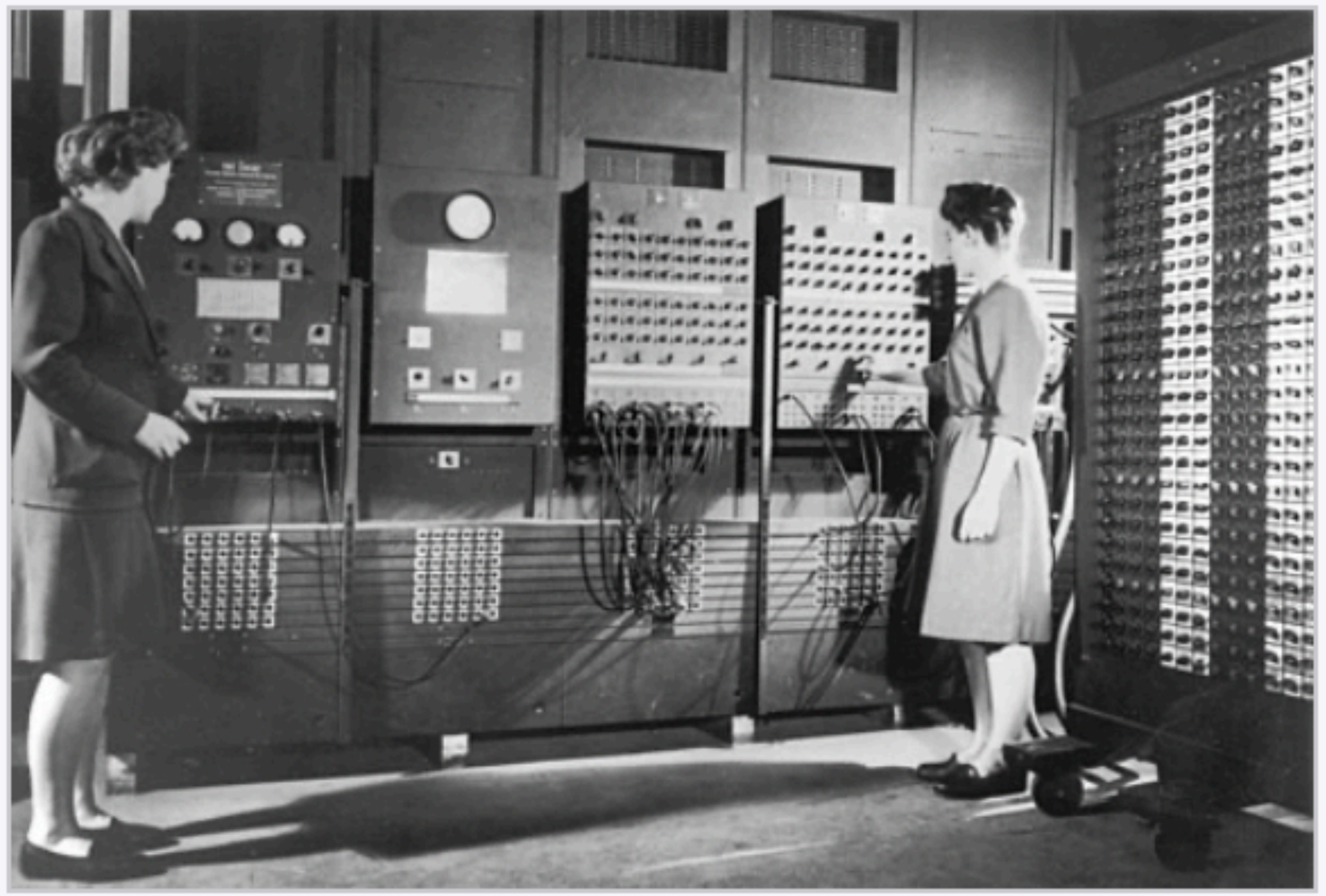


A wartime picture of a Bletchley Park Bombe

A Tale of Two Zero-Shot Problem Solvers

Modern Computers (1941-1945)

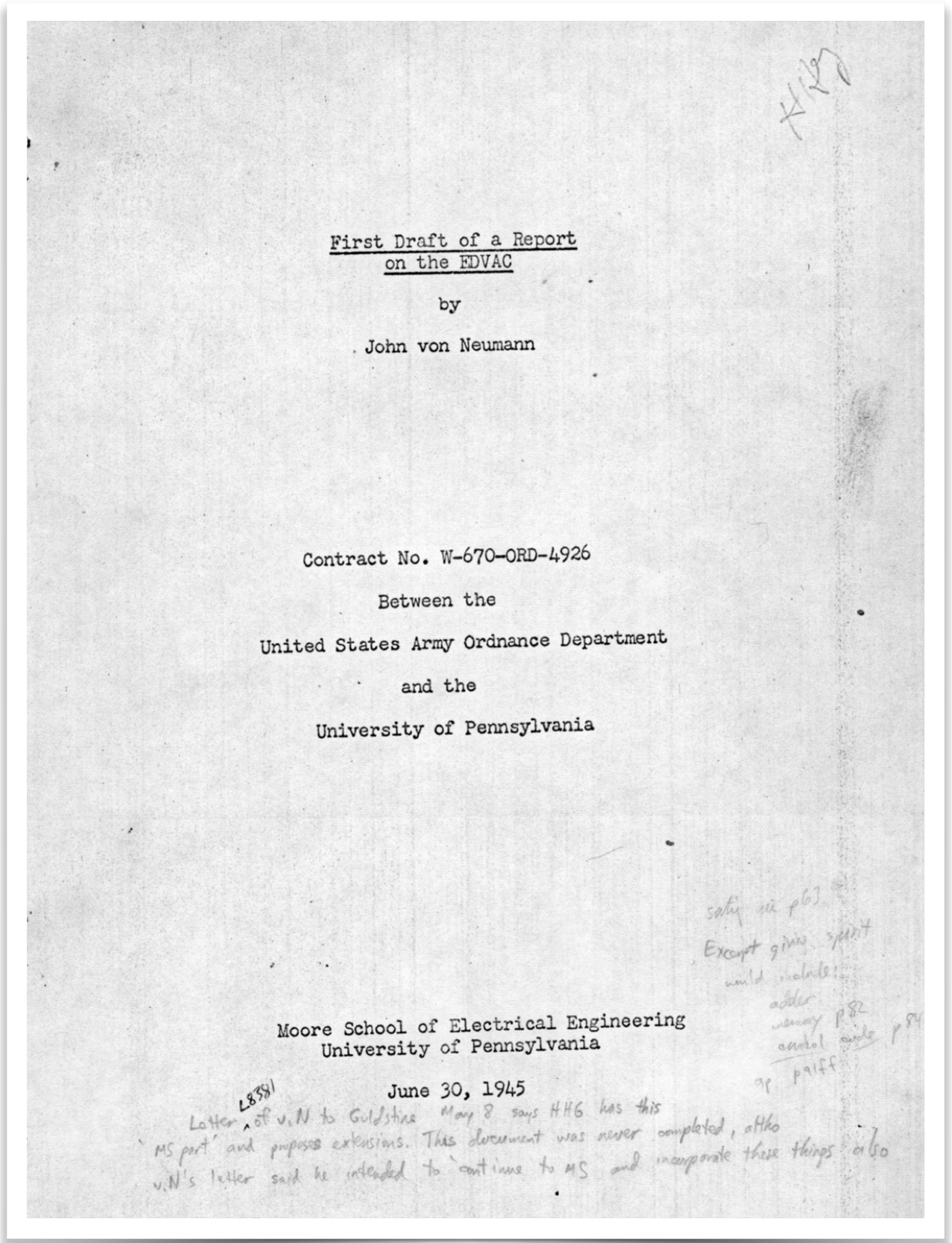
- **Hardwired Programmable Computer**
- ENIAC
- Collection of different arithmetic units and switches
- Reprogram -> Manual Rewire
- Program is still part of the hardware


Programmers Betty Jean Jennings (left) and Fran Bilas (right) operating ENIAC's main control panel at the Moore School of Electrical Engineering, c. 1945 (U.S. Army photo from the archives of the ARL Technical Library)

A Tale of Two Zero-Shot Problem Solvers

Modern Computers (1945)

- **Stored-Program Computer**
- EDVAC
- von Neumann architecture
- Represent programs as data rather than as wiring setups
- Allowed computers to read and solve arbitrary programs for which they were not set up by wiring



[illegible]

First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

satiy see p63

Except giving spirit
would include:
adder
memory p82
control code p84
ap p91ff

Moore School of Electrical Engineering
University of Pennsylvania

June 30, 1945

L8381

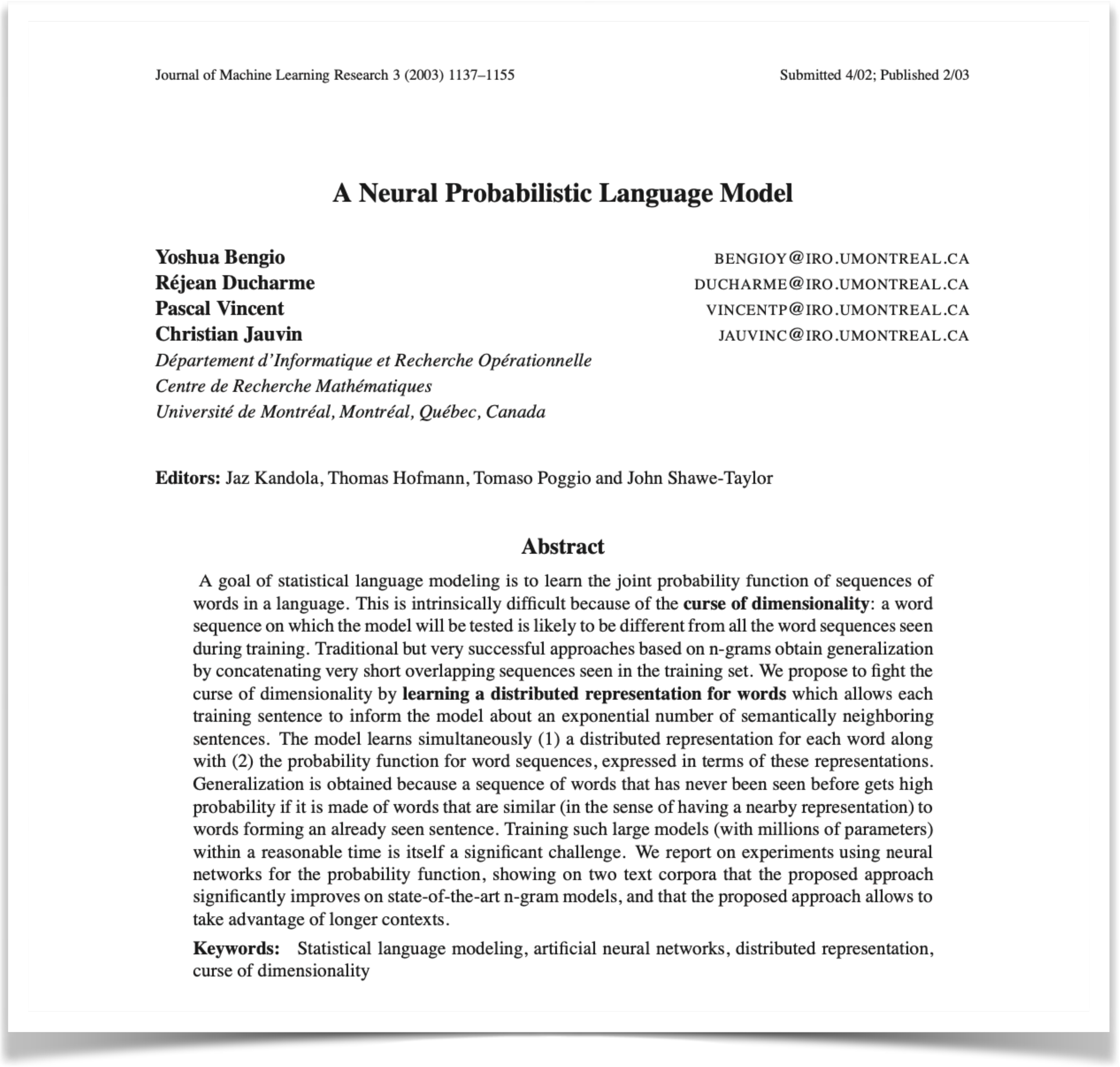
Letter of v.N to Goldstine May 8 says HHG has this
'MS part' and proposes extensions. This document was never completed, altho
v.N's letter said he intended to 'continue to MS and incorporate these things also

A Tale of Two Zero-Shot Problem Solvers

Large Language Models (2002)

- Simplest Component (logic gate of LLMs)

$$p(\mathbf{t}) = p(t_1)\prod_{i=2}^{N} p(t_i \,|\, t_1, \cdots, t_{i-1})$$



Journal of Machine Learning Research 3 (2003) 1137–1155 Submitted 4/02; Published 2/03

A Neural Probabilistic Language Model

Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Réjean Ducharme DUCHARME@IRO.UMONTREAL.CA
Pascal Vincent VINCENTP@IRO.UMONTREAL.CA
Christian Jauvin JAUVINC@IRO.UMONTREAL.CA
Département d'Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal, Montréal, Québec, Canada

Editors: Jaz Kandola, Thomas Hofmann, Tomaso Poggio and John Shawe-Taylor

Abstract

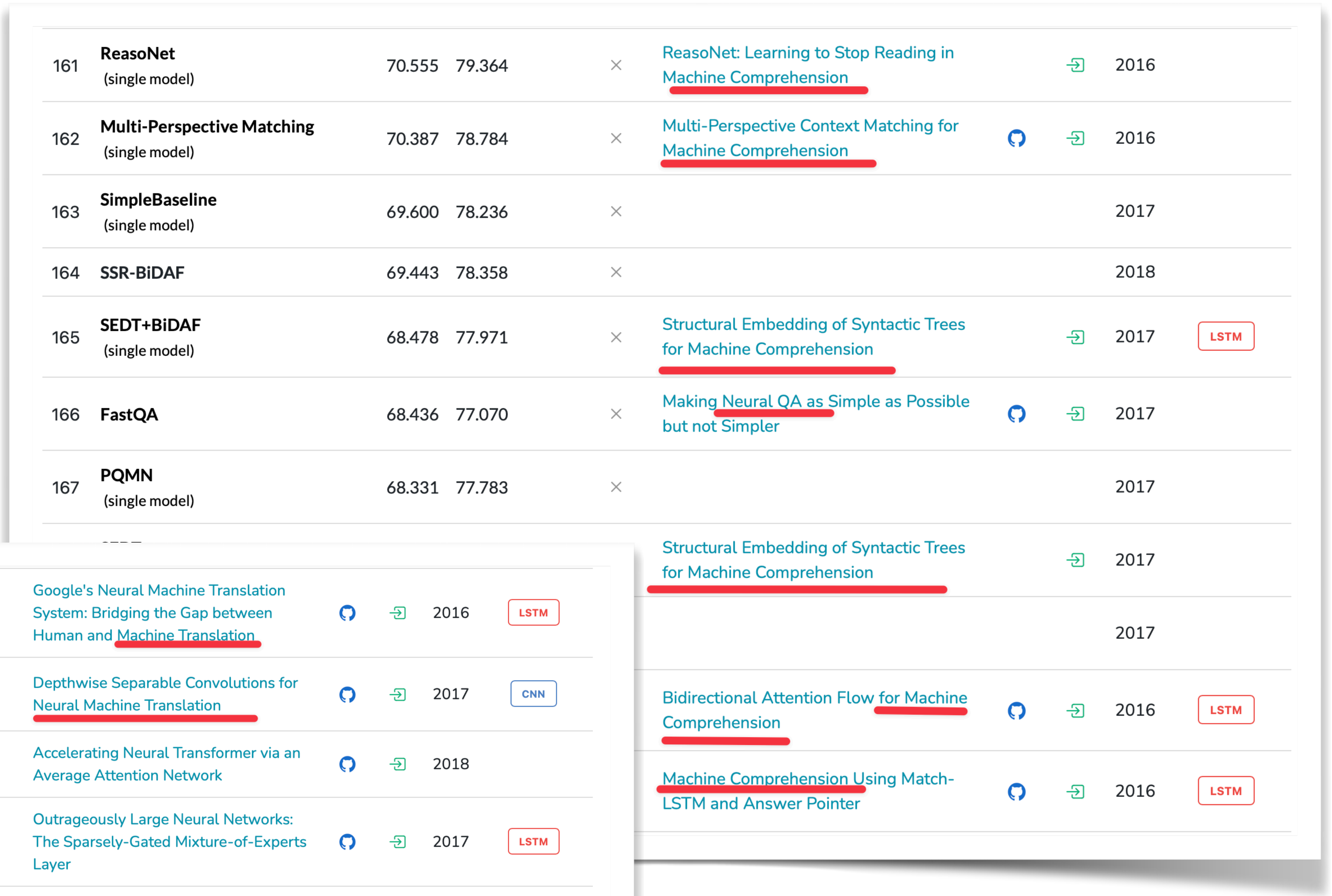
A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the **curse of dimensionality**: a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training. Traditional but very successful approaches based on n-grams obtain generalization by concatenating very short overlapping sequences seen in the training set. We propose to fight the curse of dimensionality by **learning a distributed representation for words** which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously (1) a distributed representation for each word along with (2) the probability function for word sequences, expressed in terms of these representations. Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence. Training such large models (with millions of parameters) within a reasonable time is itself a significant challenge. We report on experiments using neural networks for the probability function, showing on two text corpora that the proposed approach significantly improves on state-of-the-art n-gram models, and that the proposed approach allows to take advantage of longer contexts.

Keywords: Statistical language modeling, artificial neural networks, distributed representation, curse of dimensionality

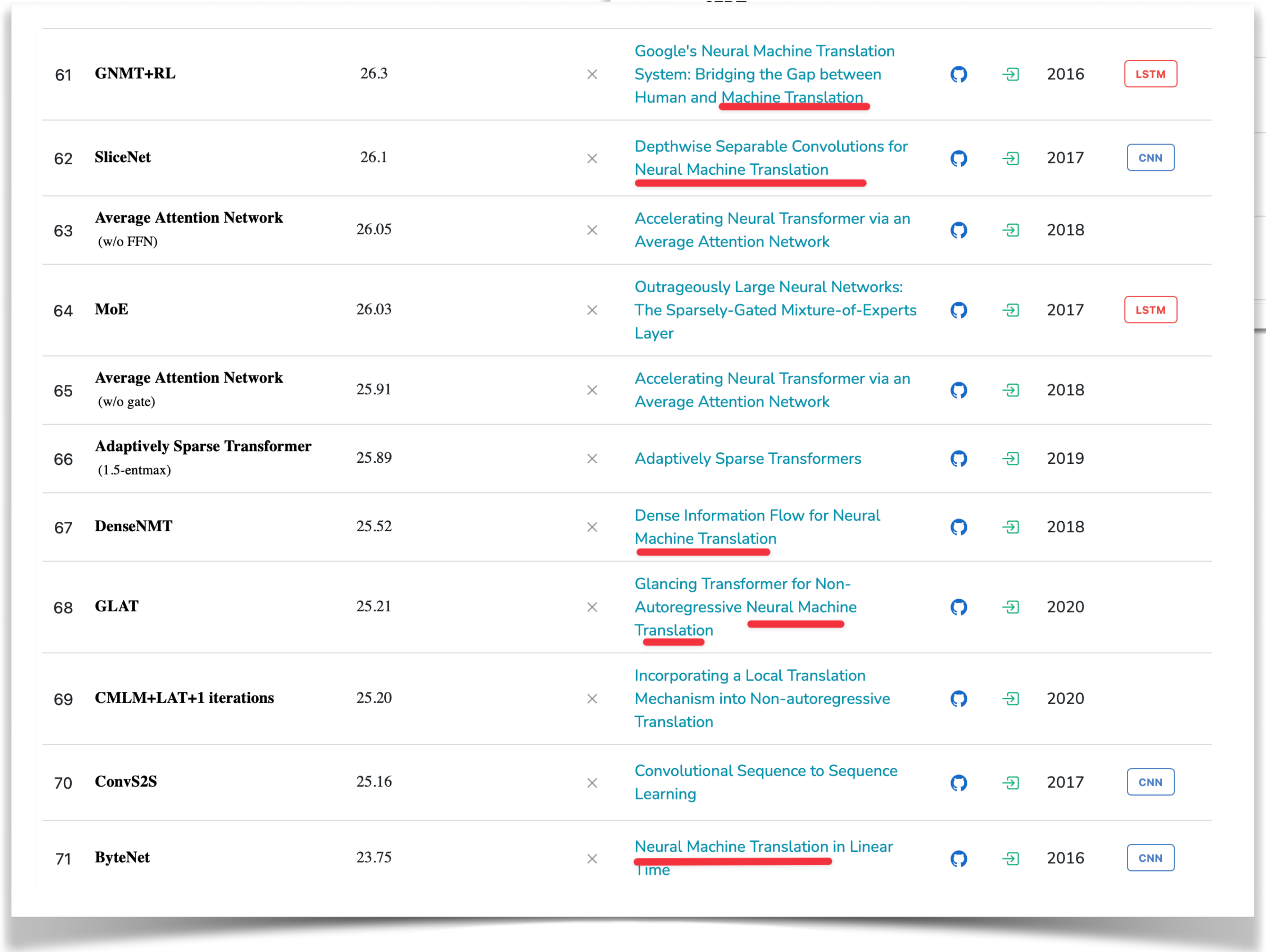
A Tale of Two Zero-Shot Problem Solvers

Large Language Models (2002-2017)

- **One model for one NLP task**, e.g.:
- $p(\text{answer} \mid \text{question})$
- $p(\text{Chinese} \mid \text{English})$



161	**ReasoNet** (single model)	70.555	79.364	×	ReasoNet: Learning to Stop Reading in Machine Comprehension	2016	
162	**Multi-Perspective Matching** (single model)	70.387	78.784	×	Multi-Perspective Context Matching for Machine Comprehension	2016	
163	**SimpleBaseline** (single model)	69.600	78.236	×		2017	
164	**SSR-BiDAF**	69.443	78.358	×		2018	
165	**SEDT+BiDAF** (single model)	68.478	77.971	×	Structural Embedding of Syntactic Trees for Machine Comprehension	2017	LSTM
166	**FastQA**	68.436	77.070	×	Making Neural QA as Simple as Possible but not Simpler	2017	
167	**PQMN** (single model)	68.331	77.783	×		2017	
					Structural Embedding of Syntactic Trees for Machine Comprehension	2017	
						2017	
					Bidirectional Attention Flow for Machine Comprehension	2016	LSTM
					Machine Comprehension Using Match-LSTM and Answer Pointer	2016	LSTM

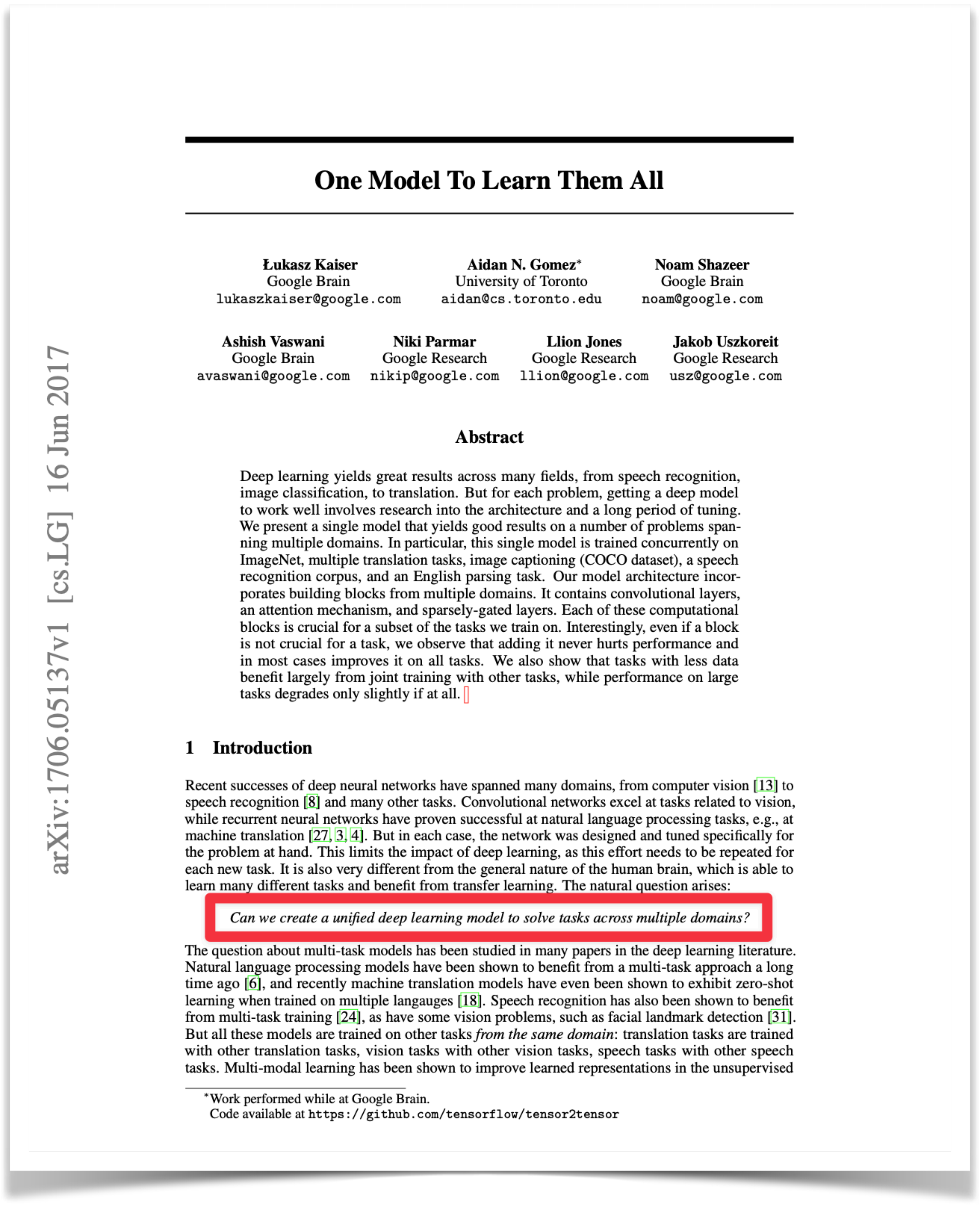


61	**GNMT+RL**	26.3	×	Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation	2016	LSTM
62	**SliceNet**	26.1	×	Depthwise Separable Convolutions for Neural Machine Translation	2017	CNN
63	**Average Attention Network** (w/o FFN)	26.05	×	Accelerating Neural Transformer via an Average Attention Network	2018	
64	**MoE**	26.03	×	Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer	2017	LSTM
65	**Average Attention Network** (w/o gate)	25.91	×	Accelerating Neural Transformer via an Average Attention Network	2018	
66	**Adaptively Sparse Transformer** (1.5-entmax)	25.89	×	Adaptively Sparse Transformers	2019	
67	**DenseNMT**	25.52	×	Dense Information Flow for Neural Machine Translation	2018	
68	**GLAT**	25.21	×	Glancing Transformer for Non-Autoregressive Neural Machine Translation	2020	
69	**CMLM+LAT+1 iterations**	25.20	×	Incorporating a Local Translation Mechanism into Non-autoregressive Translation	2020	
70	**ConvS2S**	25.16	×	Convolutional Sequence to Sequence Learning	2017	CNN
71	**ByteNet**	23.75	×	Neural Machine Translation in Linear Time	2016	CNN

A Tale of Two Zero-Shot Problem Solvers

Large Language Models (2017)

- **Solved task specified by a special tag,** e.g., <QA>, <MT>
- $p_{\text{task}}(\text{output} \mid \text{input})$



One Model To Learn Them All

Łukasz Kaiser
Google Brain
lukaszkaiser@google.com

Aidan N. Gomez*
University of Toronto
aidan@cs.toronto.edu

Noam Shazeer
Google Brain
noam@google.com

Ashish Vaswani
Google Brain
avaswani@google.com

Niki Parmar
Google Research
nikip@google.com

Llion Jones
Google Research
llion@google.com

Jakob Uszkoreit
Google Research
usz@google.com

Abstract

Deep learning yields great results across many fields, from speech recognition, image classification, to translation. But for each problem, getting a deep model to work well involves research into the architecture and a long period of tuning. We present a single model that yields good results on a number of problems spanning multiple domains. In particular, this single model is trained concurrently on ImageNet, multiple translation tasks, image captioning (COCO dataset), a speech recognition corpus, and an English parsing task. Our model architecture incorporates building blocks from multiple domains. It contains convolutional layers, an attention mechanism, and sparsely-gated layers. Each of these computational blocks is crucial for a subset of the tasks we train on. Interestingly, even if a block is not crucial for a task, we observe that adding it never hurts performance and in most cases improves it on all tasks. We also show that tasks with less data benefit largely from joint training with other tasks, while performance on large tasks degrades only slightly if at all.

1 Introduction

Recent successes of deep neural networks have spanned many domains, from computer vision [13] to speech recognition [8] and many other tasks. Convolutional networks excel at tasks related to vision, while recurrent neural networks have proven successful at natural language processing tasks, e.g., at machine translation [27, 3, 4]. But in each case, the network was designed and tuned specifically for the problem at hand. This limits the impact of deep learning, as this effort needs to be repeated for each new task. It is also very different from the general nature of the human brain, which is able to learn many different tasks and benefit from transfer learning. The natural question arises:

Can we create a unified deep learning model to solve tasks across multiple domains?

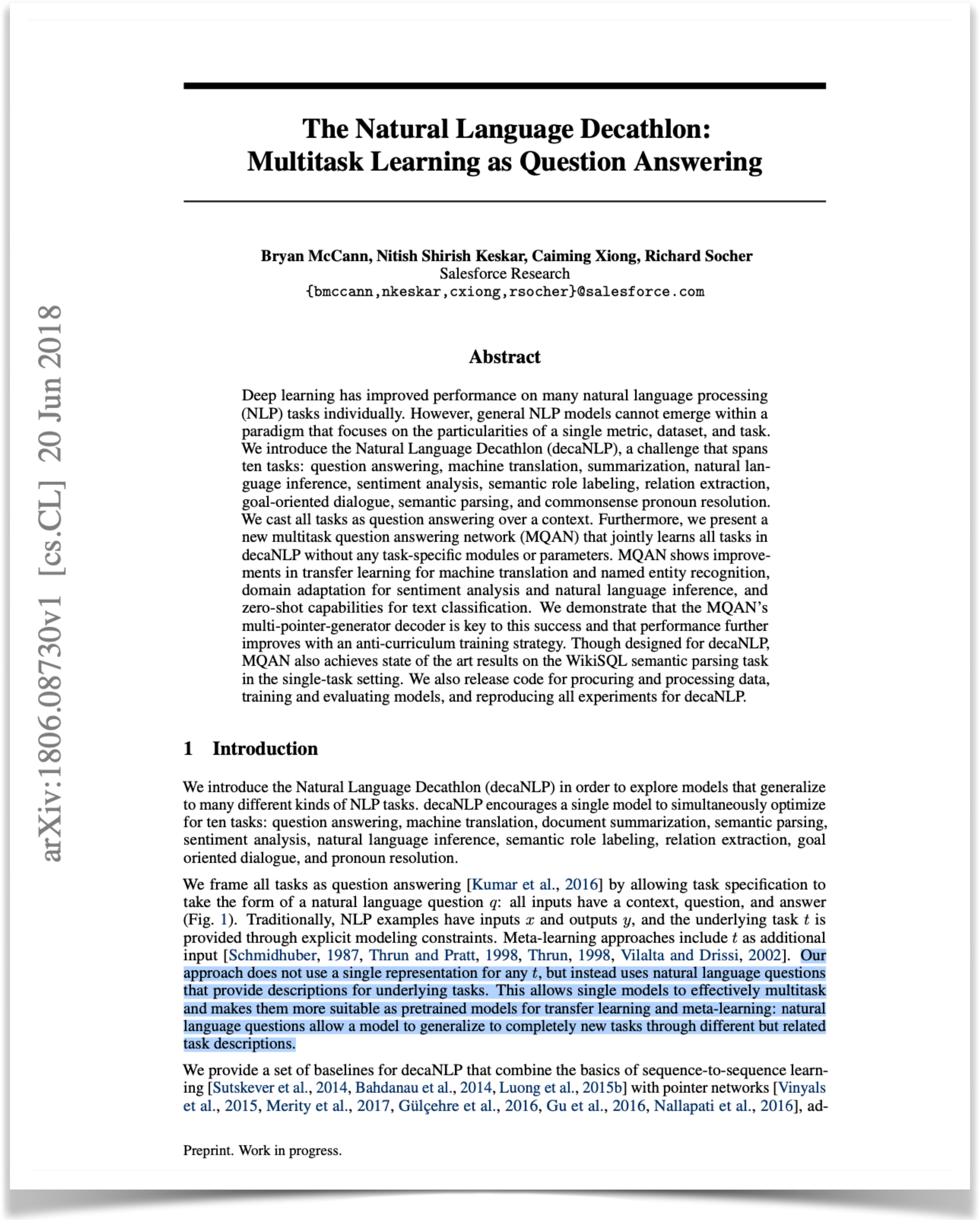
The question about multi-task models has been studied in many papers in the deep learning literature. Natural language processing models have been shown to benefit from a multi-task approach a long time ago [6], and recently machine translation models have even been shown to exhibit zero-shot learning when trained on multiple langauges [18]. Speech recognition has also been shown to benefit from multi-task training [24], as have some vision problems, such as facial landmark detection [31]. But all these models are trained on other tasks *from the same domain*: translation tasks are trained with other translation tasks, vision tasks with other vision tasks, speech tasks with other speech tasks. Multi-modal learning has been shown to improve learned representations in the unsupervised

*Work performed while at Google Brain.
Code available at `https://github.com/tensorflow/tensor2tensor`

A Tale of Two Zero-Shot Problem Solvers

Large Language Models (2018)

- **Task, Input, and Output represented in the same format**
  (i.e., natural language)
- $p(\text{output} \mid \text{task}, \text{input})$
- Cf. von Neumann architecture



arXiv:1806.08730v1 [cs.CL] 20 Jun 2018

**The Natural Language Decathlon:
Multitask Learning as Question Answering**

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, Richard Socher
Salesforce Research
{bmccann,nkeskar,cxiong,rsocher}@salesforce.com

Abstract

Deep learning has improved performance on many natural language processing (NLP) tasks individually. However, general NLP models cannot emerge within a paradigm that focuses on the particularities of a single metric, dataset, and task. We introduce the Natural Language Decathlon (decaNLP), a challenge that spans ten tasks: question answering, machine translation, summarization, natural language inference, sentiment analysis, semantic role labeling, relation extraction, goal-oriented dialogue, semantic parsing, and commonsense pronoun resolution. We cast all tasks as question answering over a context. Furthermore, we present a new multitask question answering network (MQAN) that jointly learns all tasks in decaNLP without any task-specific modules or parameters. MQAN shows improvements in transfer learning for machine translation and named entity recognition, domain adaptation for sentiment analysis and natural language inference, and zero-shot capabilities for text classification. We demonstrate that the MQAN's multi-pointer-generator decoder is key to this success and that performance further improves with an anti-curriculum training strategy. Though designed for decaNLP, MQAN also achieves state of the art results on the WikiSQL semantic parsing task in the single-task setting. We also release code for procuring and processing data, training and evaluating models, and reproducing all experiments for decaNLP.

1 Introduction

We introduce the Natural Language Decathlon (decaNLP) in order to explore models that generalize to many different kinds of NLP tasks. decaNLP encourages a single model to simultaneously optimize for ten tasks: question answering, machine translation, document summarization, semantic parsing, sentiment analysis, natural language inference, semantic role labeling, relation extraction, goal oriented dialogue, and pronoun resolution.

We frame all tasks as question answering [Kumar et al., 2016] by allowing task specification to take the form of a natural language question q: all inputs have a context, question, and answer (Fig. 1). Traditionally, NLP examples have inputs x and outputs y, and the underlying task t is provided through explicit modeling constraints. Meta-learning approaches include t as additional input [Schmidhuber, 1987, Thrun and Pratt, 1998, Thrun, 1998, Vilalta and Drissi, 2002]. Our approach does not use a single representation for any t, but instead uses natural language questions that provide descriptions for underlying tasks. This allows single models to effectively multitask and makes them more suitable as pretrained models for transfer learning and meta-learning: natural language questions allow a model to generalize to completely new tasks through different but related task descriptions.

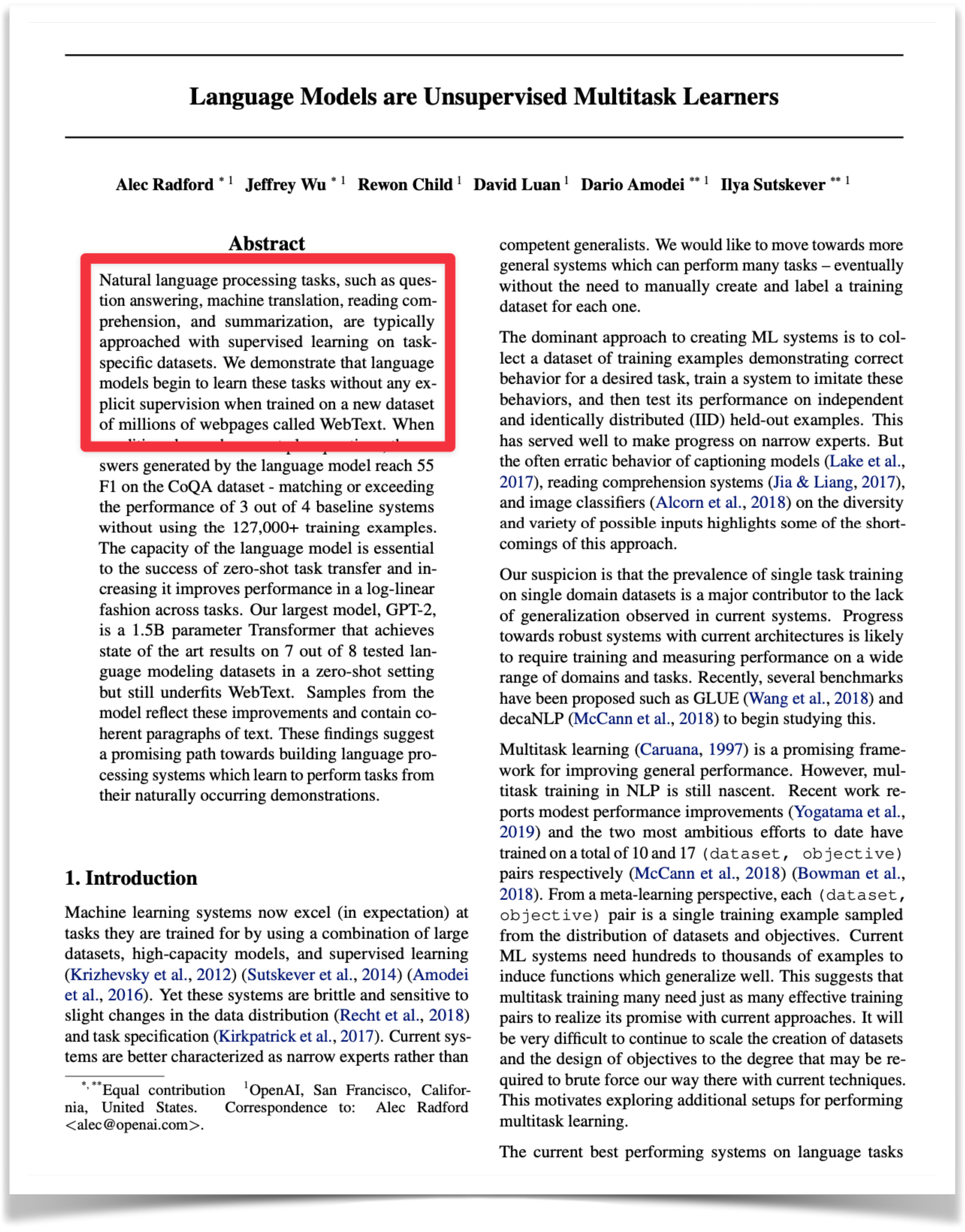
We provide a set of baselines for decaNLP that combine the basics of sequence-to-sequence learning [Sutskever et al., 2014, Bahdanau et al., 2014, Luong et al., 2015b] with pointer networks [Vinyals et al., 2015, Merity et al., 2017, Gülçehre et al., 2016, Gu et al., 2016, Nallapati et al., 2016], ad-

Preprint. Work in progress.

A Tale of Two Zero-Shot Problem Solvers

Large Language Models (2019)

- **Language Modeling (GPT-2)**
- $p(\text{response} \mid \text{prompt})$



Language Models are Unsupervised Multitask Learners

Alec Radford [*] [1] Jeffrey Wu [*] [1] Rewon Child [1] David Luan [1] Dario Amodei [**] [1] Ilya Sutskever [**] [1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When [illegible] swers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

1. Introduction

Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Krizhevsky et al., 2012) (Sutskever et al., 2014) (Amodei et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Recht et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems are better characterized as narrow experts rather than competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

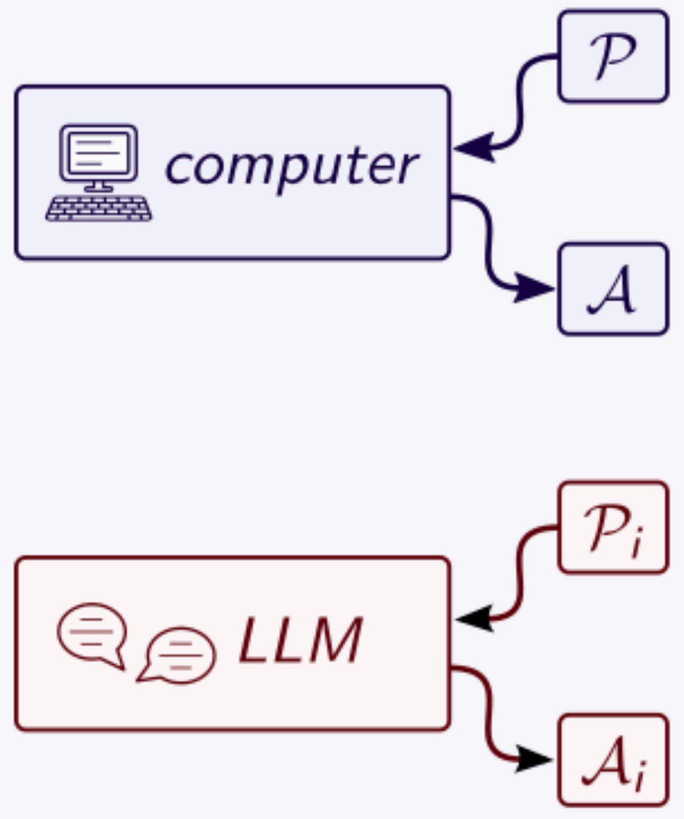
Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 `(dataset, objective)` pairs respectively (McCann et al., 2018) (Bowman et al., 2018). From a meta-learning perspective, each `(dataset, objective)` pair is a single training example sampled from the distribution of datasets and objectives. Current ML systems need hundreds to thousands of examples to induce functions which generalize well. This suggests that multitask training many need just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be required to brute force our way there with current techniques. This motivates exploring additional setups for performing multitask learning.

The current best performing systems on language tasks

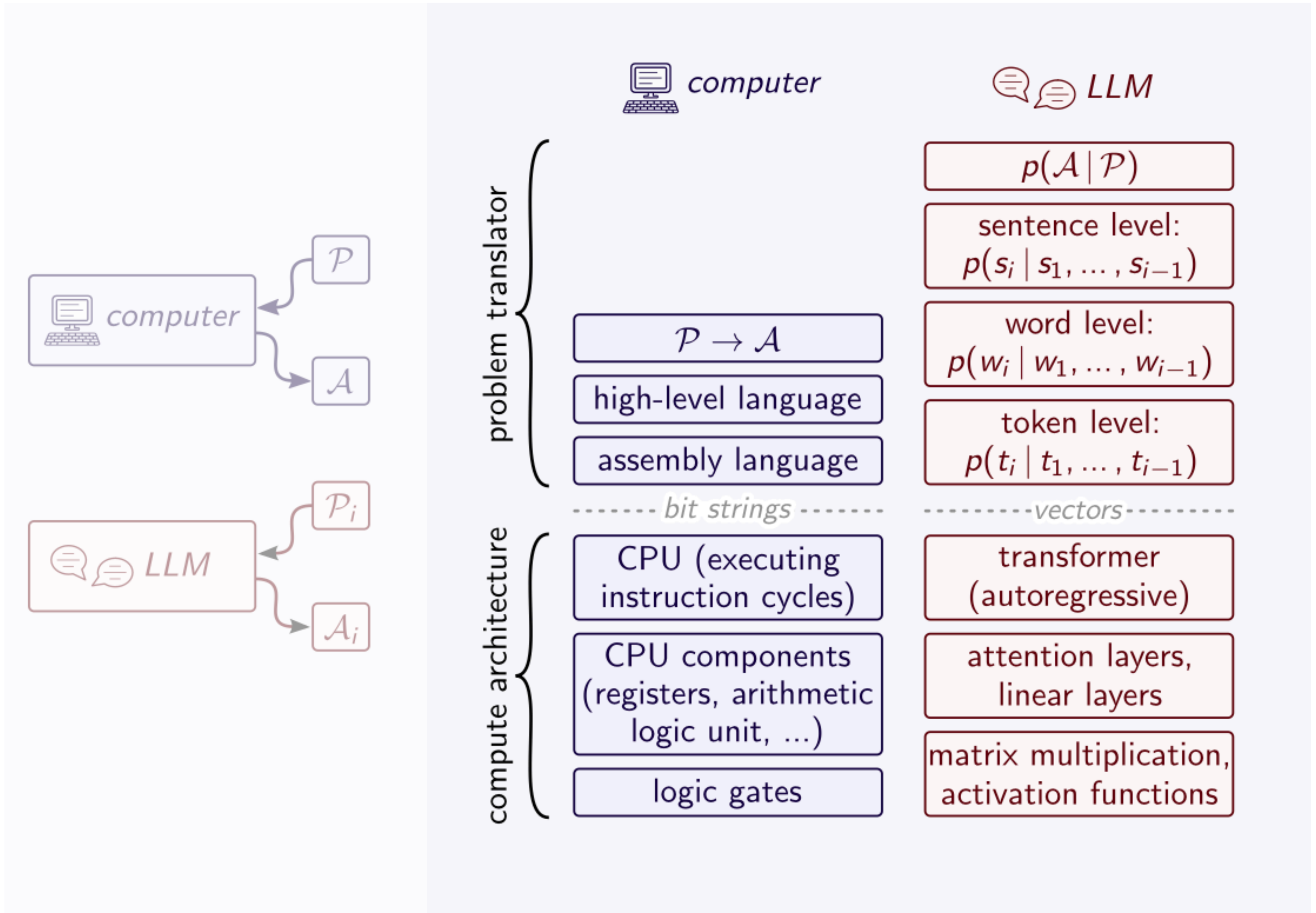
[*], [**] Equal contribution [1] OpenAI, San Francisco, California, United States. Correspondence to: Alec Radford <alec@openai.com>.

Connecting the Dots



(a) Framework

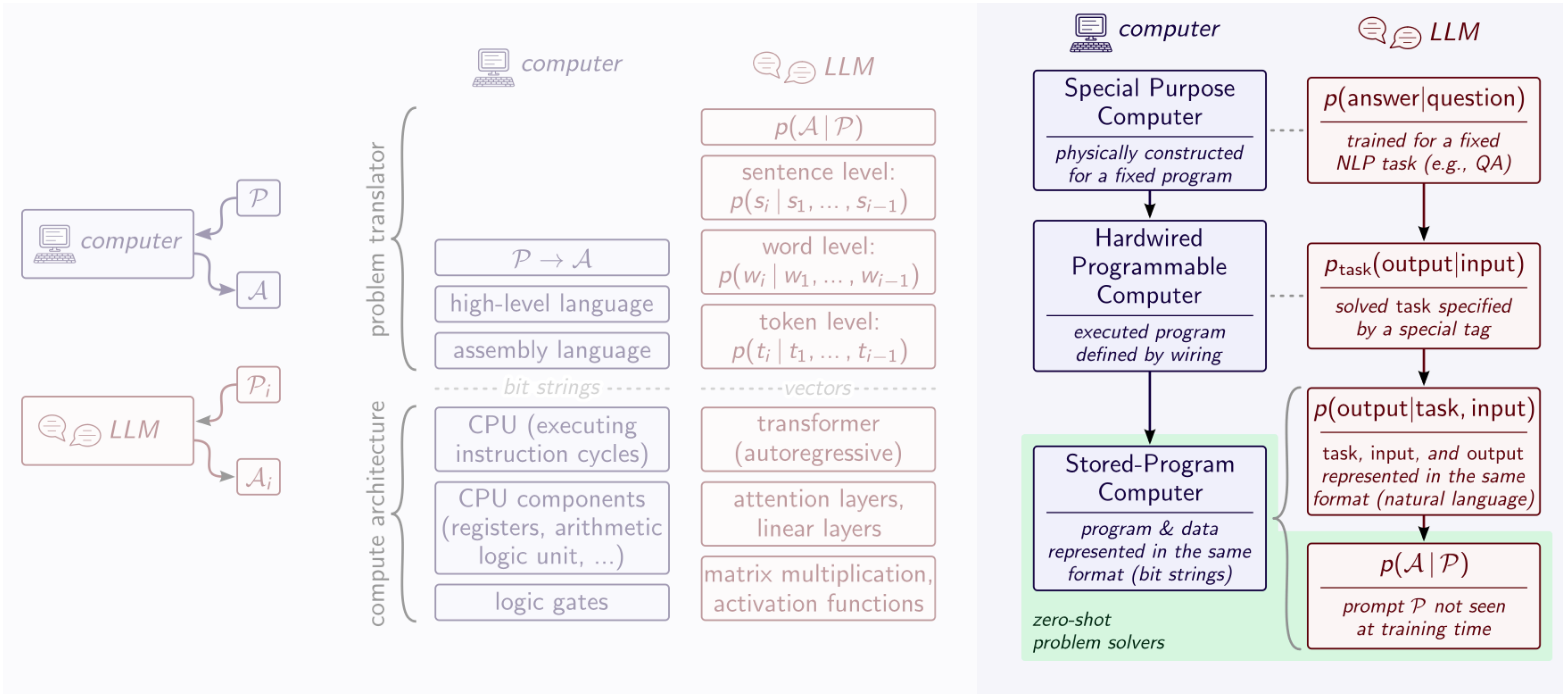
Connecting the Dots



computer
$\mathcal{P}$
$\mathcal{A}$
LLM
$\mathcal{P}_i$
$\mathcal{A}_i$
computer
LLM
problem translator
compute architecture
$\mathcal{P} \to \mathcal{A}$
high-level language
assembly language
bit strings
CPU (executing instruction cycles)
CPU components (registers, arithmetic logic unit, ...)
logic gates
$p(\mathcal{A} \mid \mathcal{P})$
sentence level: $p(s_i \mid s_1, \dots, s_{i-1})$
word level: $p(w_i \mid w_1, \dots, w_{i-1})$
token level: $p(t_i \mid t_1, \dots, t_{i-1})$
vectors
transformer (autoregressive)
attention layers, linear layers
matrix multiplication, activation functions

(a) Framework (b) Levels of abstraction

Connecting the Dots



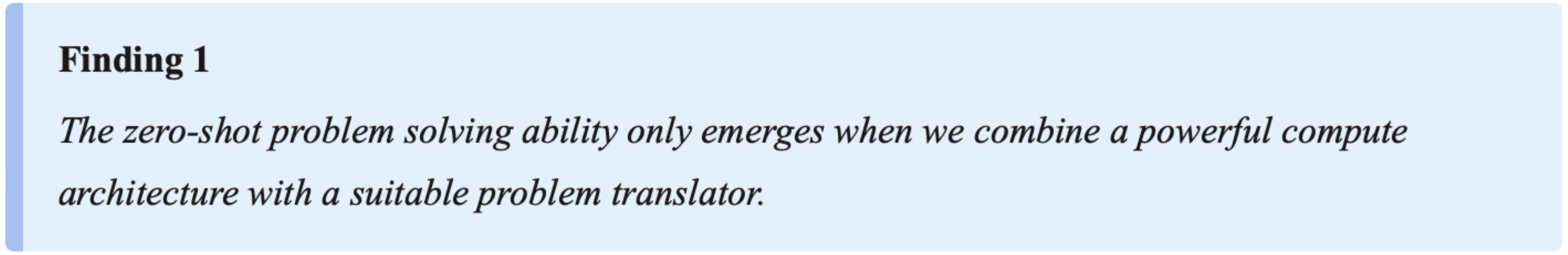
computer
$\mathcal{P}$
$\mathcal{A}$
LLM
$\mathcal{P}_i$
$\mathcal{A}_i$
computer
LLM
problem translator
compute architecture
$p(\mathcal{A} \mid \mathcal{P})$
sentence level: $p(s_i \mid s_1, \ldots, s_{i-1})$
word level: $p(w_i \mid w_1, \ldots, w_{i-1})$
token level: $p(t_i \mid t_1, \ldots, t_{i-1})$
$\mathcal{P} \rightarrow \mathcal{A}$
high-level language
assembly language
bit strings
vectors
CPU (executing instruction cycles)
CPU components (registers, arithmetic logic unit, ...)
logic gates
transformer (autoregressive)
attention layers, linear layers
matrix multiplication, activation functions
computer
LLM
Special Purpose Computer
physically constructed for a fixed program
Hardwired Programmable Computer
executed program defined by wiring
Stored-Program Computer
program & data represented in the same format (bit strings)
$p(\text{answer} \mid \text{question})$
trained for a fixed NLP task (e.g., QA)
$p_{\text{task}}(\text{output} \mid \text{input})$
solved task specified by a special tag
$p(\text{output} \mid \text{task, input})$
task, input, and output represented in the same format (natural language)
$p(\mathcal{A} \mid \mathcal{P})$
prompt $\mathcal{P}$ not seen at training time
zero-shot problem solvers

(a) Framework (b) Levels of abstraction (c) Paths towards universality

Fundamental Difference

- LLMs:
  - More accessible but expensive due to informal reasoning
  - Ambiguous and difficult to describe a problem precisely
- Computers:
  - A world that is formally structured and logically well-defined
  - Required to learn these formal languages in order to gain full access to computer's problem solving ability

Lessons to Learn



Finding 1

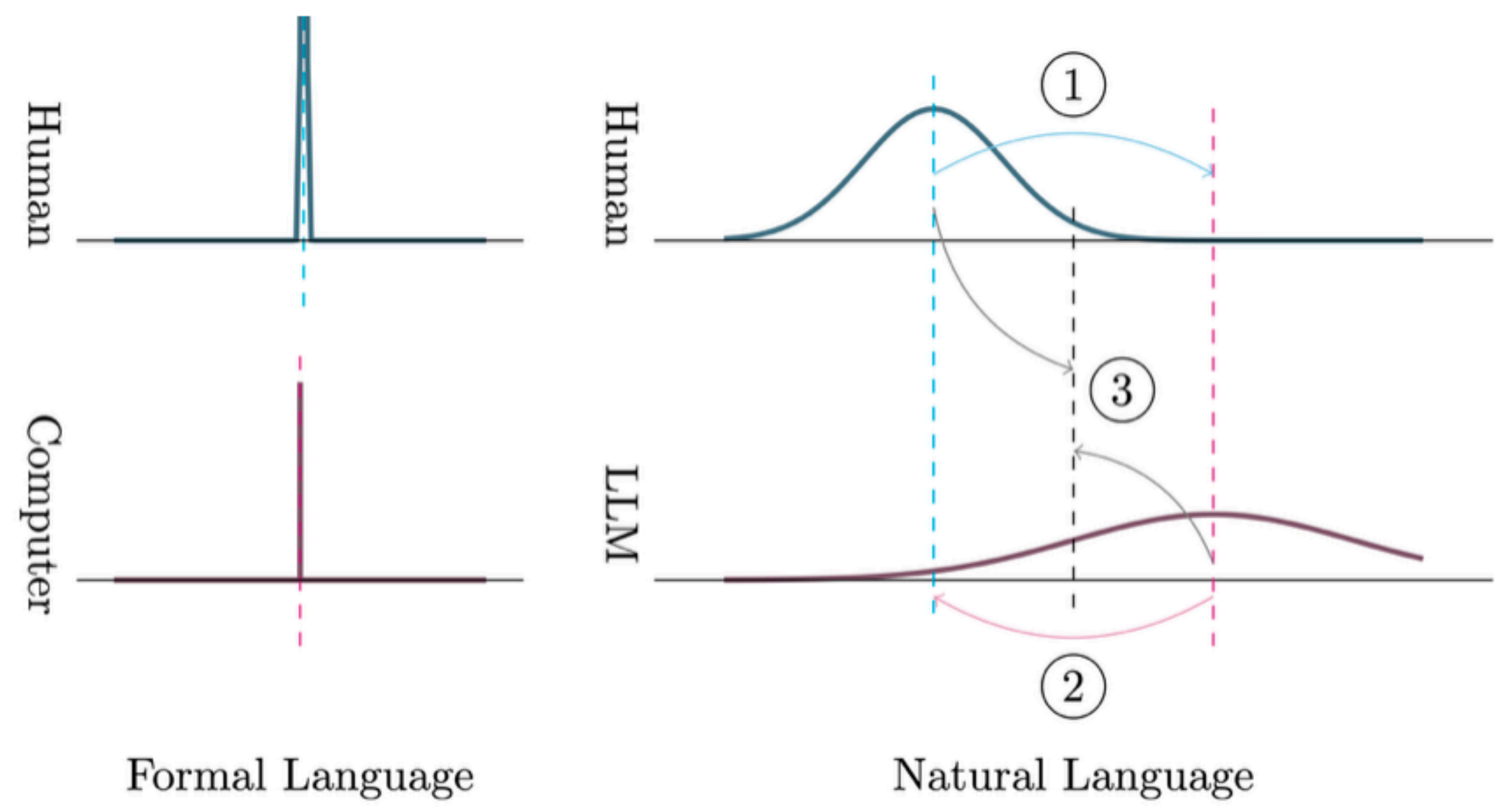
The zero-shot problem solving ability only emerges when we combine a powerful compute architecture with a suitable problem translator.

- Computational ability is not as rare a property as we might think.
- E.g., Dropping a ball onto a surface

Lessons to Learn

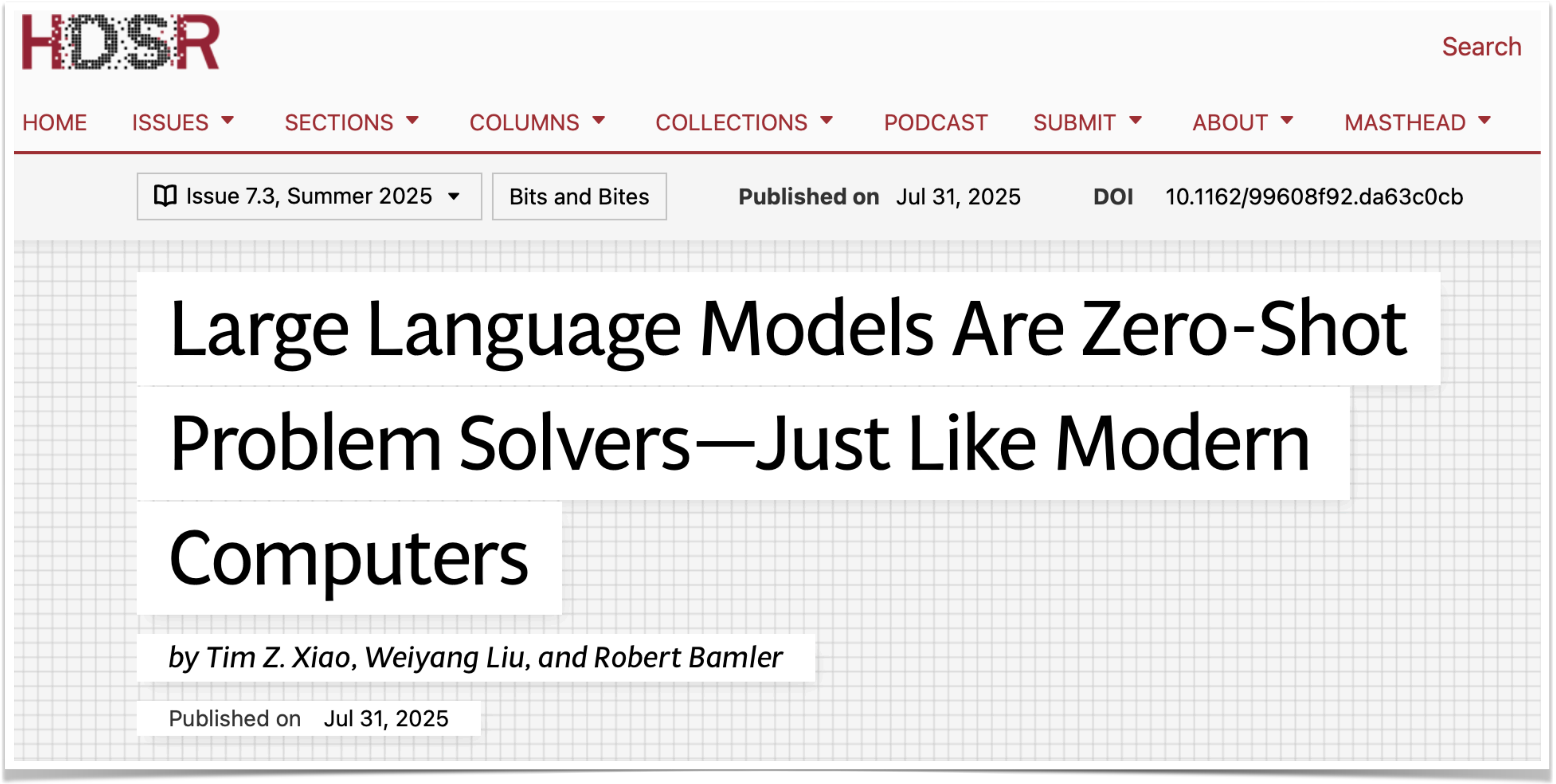
Finding 2

Minimizing the gap between the language model of LLMs and the internal language model of a user is at the core of unlocking the full problem solving ability of LLMs.



Verbalized Computing?

Q&A



Issue 7.3, Summer 2025 | Bits and Bites | **Published on** Jul 31, 2025 | **DOI** 10.1162/99608f92.da63c0cb

Large Language Models Are Zero-Shot Problem Solvers—Just Like Modern Computers

by Tim Z. Xiao, Weiyang Liu, and Robert Bamler

Published on Jul 31, 2025

https://doi.org/10.1162/99608f92.da63c0cb