Natural Language Engineering (2024), **30**, pp. 1323–1335
doi:10.1017/S1351324924000068

CAMBRIDGE UNIVERSITY PRESS

EMERGING TRENDS

Emerging trends: evaluating general purpose foundation models

Kenneth Ward Church[1] and Omar Alonso[2]

[1]Northeastern University, Boston, MA 02139, USA and [2]Amazon, Palo Alto, CA, USA
Corresponding author: Kenneth Ward Church; Email: k.church@northeastern.edu

(Received 4 November 2024; revised 4 November 2024)

Abstract

We suggest that foundation models are general purpose solutions similar to general purpose programmable microprocessors, where fine-tuning and prompt-engineering are analogous to coding for microprocessors. Evaluating general purpose solutions is not like hypothesis testing. We want to know how well the machine will perform on an unknown program with unknown inputs for unknown users with unknown budgets and unknown utility functions. This paper is based on an invited talk by John Mashey, "Lessons from SPEC," at an ACL-2021 workshop on benchmarking. Mashey started by describing Standard Performance Evaluation Corporation (SPEC), a benchmark that has had more impact than benchmarks in our field because SPEC addresses an import commercial question: which CPU should I buy? In addition, SPEC can be interpreted to show that CPUs are 50,000 faster than they were 40 years ago. It is remarkable that we can make such statements without specifying the program, users, task, dataset, etc. It would be desirable to make quantitative statements about improvements of general purpose foundation models over years/decades without specifying tasks, datasets, use cases, etc.

Keywords: GLUE; SPEC; benchmarking; general purpose computing; geometric mean

1. Lessons from SPEC

This paper[a] is based on invited talk by John Mashey, "Lessons from SPEC,"[b] at the ACL-2021 Workshop on Benchmarking: Past, Present and Future (BPPF) (Church *et al.* 2021). John Mashey[c] is very well known in Systems, but less so in Natural Language Processing.

Mashey played an important role in the creation of Standard Performance Evaluation Corporation (SPEC)[d] (Dixit 1993). SPEC has had more influence than most benchmarks in natural language because SPEC addresses a commercially important question: which computer should I buy? SPEC is the de facto standard for evaluating the performance of CPUs (and more).

Mashey's talk starts out with a discussion of the history of the SPEC benchmark and then uses some of those lessons to criticize GLUE/SuperGLUE (Wang *et al.* 2018, 2019), important benchmarks in natural language processing. His criticisms are also applicable to many/most benchmarks in fields of interest to our community: machine learning, natural language processing, information retrieval, etc.

[a]Work does not relate to the second author's position at Amazon.
[b]https://github.com/kwchurch/Benchmarking_past_present_future?tab=readme-ov-file#Mashey
[c]https://en.wikipedia.org/wiki/John_Mashey
[d]https://www.spec.org/

Table 1. It is easy to show that deep nets are becoming larger and larger over time (Church 2022), but harder to make quantitative statements about improvements in quality over time. Can we quantify the impact of this progress for typical customers?

Year	Deep Nets	Reference	Parameters (in Billions)
2016	ResNet-50	He *et al.* (2016)	0.023
2019	BERT	Devlin *et al.* (2019)	0.34
2019	GPT-2	Radford *et al.* (2019)	1.5
2020	GPT-3	Brown *et al.* (2020)	175
2022	PaLM	Chowdhery *et al.* (2023)	540

Table 2. SPECRatios have grown by a factor of 50,000 over 40 years

Interval	Growth	SPECRatio	Reasons for Slowdown
1978–1986	20%/year	1–5	
1986–2003	52%/year	5–6k	
2003–2011	23%/year	6k–32k	
2011–2015	12%/year	32k–50k	Amdahl's law limits
2015–2018	3.5%/year	50k	Moore's law ends

2. Goal: a task/Dataset-independent score of ML quality

It is common in the literature on foundation models to show that models are becoming larger and larger over time, as illustrated in Table 1. The industry is moving toward bigger models because it is widely believed that bigger models are better. That is, after programming with fine-tuning and/or prompting and tweaking hyper-parameters, users can expect to see better scores if they start with bigger foundation models (for many/most users/tasks/metrics).

But we do not have a way to quantify this consensus. It would be desirable to produce a chart that shows gains over the past decade relative to a baseline such as BERT (Devlin *et al.* 2019), or even better, human performance. ML Commons[e] is making progress in this direction, though thus far, much of the effort has been focused more on measuring gains in execution speed for a particular task and a particular dataset, as opposed to measuring gains in quality (scores) that users can expect for their tasks and their workloads.

In short, we need a way to measure gains in quality over the long term (decades). This measurement should help users decide which general purpose foundation model they should use. The measurement should be credible over a wide range of tasks and datasets since general purpose foundation models will be used by many users for many use cases.

We will show, when we discuss Table 2, that SPEC makes it possible to make quantitative statements about performance improvements of general purpose computers over decades. The goal is to make similar statements for general purpose foundation models.

[e] https://mlcommons.org/benchmarks/inference-datacenter/

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

3. Evaluating general purpose solutions

SPEC was designed to evaluate CPUs, but CPUs are designed for general purpose computing. The challenge is how to evaluate a machine that can do anything for anyone, or at least many different things for many different users with many different needs under many different scenarios. How can we evaluate a general purpose solution without addressing questions such as these:

- Who is going to use the machine to do what?
- What is a typical workload?
- What is a typical user?
- What is a typical use case?
- What is a typical scenario?

Evaluating general purpose solutions is not like hypothesis testing. We want to know how well the machine will perform on an unknown program with unknown inputs for unknown users with unknown budgets and unknown utility functions. If we were to evaluate a search engine, we could hope to estimate typical workloads by sampling the logs. It is standard practice in many fields to estimate workloads: databases (Zhang *et al.* 2018), networking (Calzarossa *et al.* 2016), and web search (Broder 2002). But how do we sample typical workloads for a general purpose machine?

Clearly, CPUs are more general purpose than search engines and databases. By design, CPUs are jack of all trades, master of none.[f] If we knew what the CPU was going to be used for, we could design a special purpose solution that would work better on that task, but that is missing the point for general purpose solutions; the point of general purpose solutions is to be general purpose.

Why did chip manufacturers decide to build general purpose chips? When Intel was starting out, it was very expensive to design a chip. Intel had a contract in 1969 from Busicom,[g] a calculator company, to build a twelve-chip set for a desktop calculator. Instead of building 12 chips, Intel built a single chip, the 4004 programmable microprocessor (Faggin 2018). This microprocessor could then be programmed to perform all 12 functions for this customer, as well as many other functions for many other customers.[h]

The idea of general purpose computing, of course, was well known well before the microprocessor (Turing 1936), but VLSI technology drastically reduced variable costs. While the fixed costs to design the first instance of a chip remain high, with VLSI technology, the variable costs to print each addition instance became a round-off error compared to previous technologies. Chip suppliers quickly discovered that it was more profitable to design a single general purpose chip that could be mass produced and sold to a mass market than to design many special purpose chips for many smaller markets. In this way, design costs could be amortized over many customers and many use cases.

We are beginning to encounter similar challenges and opportunities in our field. Foundation models are becoming so expensive to train that we should think of them as general purpose solutions for many customers and many use cases. We think of fine-tuning and prompt-engineering as methods of programming foundation models, not all that different from programming the Intel 4004 microprocessor. Under this view, evaluation of foundation model should be viewed more like evaluating general purpose solutions as opposed to the way we have traditionally evaluated special purpose solutions for more specific tasks such as web search.

[f] https://en.wikipedia.org/wiki/Jack_of_all_trades
[g] https://en.wikipedia.org/wiki/Busicom
[h] https://www.intel.com/content/www/us/en/history/virtual-vault/articles/the-intel-4004.html

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

4. SPEC use case: what should I buy?

Customers were asking for a simple answer to a simple question: what should they buy? They want a single number, not a long complicated story such as a precision/recall curve. Consumers are not going to read a thoughtful review like you might see in consumer reports. SPEC originally proposed two numbers, but even that was too complicated. The market demanded a simple answer to their simple question; customers want a simple credible number that is easy to interpret and easy to use for comparing alternatives.

A simple credible number is also good for suppliers. Previous benchmarks such as Whetstone[i] and Dhrystone[j] encouraged pointless exercises such as designing hardware features to do well on the benchmarks in ways that will not generalize well to real workloads from real customers.

5. A benchmark walked into a bar

Mashey's description of the history of SPEC (below) is entertaining, but there is a serious point to his story. The computer industry owes a huge debt of gratitude to SPEC. If this tech-savvy bar owner had not intervened as he did, then SPEC might not have happened. Without SPEC, it would have been more difficult for customers to decide what to buy, and suppliers would have spent many pointless hours gaming pointless benchmarks in counter-productive ways. Unfortunately, our field has not benefited from such an intervention. As a result, our leaderboards encourage pointless SOTA-chasing (Church and Kordoni 2022), and our users are having difficulty differentiating one foundation model from another.

[A tech-savvy bar owner in Silicon Valley said], *'look, if you don't like it, give me something better I can use'. . .*
and that's how we ended up in his bar.
We all hated the fact that. . . Dhrystone plagued us because all of us had had marketing folks come in and said 'can't you add an extra instruction that will do Dhrystone really well.'
'No, no, stop; we don't want that.'
We sort of compared notes, and we found we all used some of the same benchmarks, close, but not quite the same inputs.
So we all used the GCC, okay, all right, but with different inputs. We all used the spice circuit simulator, but with different inputs. So we all had to run stuff. . .
It was just a waste of time.
So we thought it would be a good idea. . ., if we did something more consistent. . .
So that's really how SPEC got started in that bar.[k]

6. Benchmarks before SPEC

As mentioned above, there were a number of benchmarks before SPEC such as Whetstone and Dhrystone, simple programs designed to test features such as floating point, integer arithmetic, and string operations. The assessment in textbooks (Hennessy and Patterson 2012) (section 1.8) is that "benchmarks based on running programs that are much simpler than a real application have led to performance pitfalls." Examples such as Whetstone and Dhrystone "are discredited today, usually because the compiler writer and architect can conspire to make the computer appear faster on these stand-in programs than on real applications." Nevertheless, "Dhrystone is still the most widely quoted benchmark for embedded processors."

[i] https://www.netlib.org/benchmark/whetstone.c
[j] https://en.wikipedia.org/wiki/Dhrystone
[k] https://www.youtube.com/watch?v=koSuxS3QFDk

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

Table 3. SPEC CINT92 suite (from Table 2 in Giladi and Ahituv (1995))

Code	App Area	Lines	Remarks
gcc	Compiler	87,800	CNU C Compiler V1.35, compiles 76 soruces, 10% I/O
Espresso	Logic Design	14,800	PALs generation tool, heuristic minimization, little paging
Li	Interpreter	7700	Lisp interpreter (XLIST 16), solves 8-queens problem using recursive backtracking, many jumps/loops
Eqntott	Logic design	3500	Creates truth tables; > 95% of time in qsort
Compress	Data compression	1500	Compress/decompress 1 MB file 20 times using adaptive Lempel-Ziv coding
Sc	Spreadsheet	8500	Spreadsheet app based on Unix "curses"

7. Interpretation of SPECRatios

SPECRatios were designed to be interpretable. The baseline SPECRatio is set to 1 for 1978 VAX 11/780. All other SPECRatios can be interpreted as multiplicative speedups over this baseline. For example, the 1986 VAX 8700 was assigned a SPECRatio of 5, which means that users can expect the VAX 8700 to be 5 times faster than baseline. It is remarkable that we can make such a statement given how little we know. Who are the users? What do they plan to do with these machines?

Table 2 shows SPECRatios have grown dramatically by a factor of 50,000 over 40 years.[l] Most benchmarks in our field are not interpretable in this way. It would be desirable to make quantitative statements about improvements of general purpose foundation models over years without having to specify tasks, datasets, use cases, etc. Mashey suggests part of the problem is related to how we compute averages, as will be discussed in the next section.

8. No mean feat

The title of this section was inspired by a talk that Mashey gave titled *Summarizing performance is no mean feat* . . . (Mashey 2004, 2005). His point is that we should use geometric means (GMs) instead of arithmetic means.

What are we averaging over? Many benchmarks in our field consist of a set of tasks. SPEC also aggregates scores over tasks (programs), as illustrated in Table 3. The set of programs is constantly evolving. See here[m] for a more recent set.

There are many challenges including how to come up with an appropriate set of tasks and how to keep that set up to date with advances in technology. This section will focus on how to aggregate scores over tasks to produce a SPECRatio score that can be interpreted as a speedup relative to a baseline (1978 VAX 11/780).

The discussion of GMs and SPEC in textbooks, Hennessy and Patterson (2012) (section 1.8), henceforth HPs1.8, is more accessible than the primary literature. HPs1.8 start by defining

[l]Table 2 is based on slide 9 of https://courses.grainger.illinois.edu/CS433/fa2021/slides/chapter1-part1-post-lecture.pdf; there is a similar chart in Figure 1.1 of Hennessy and Patterson (2012).

[m]https://www.spec.org/cpu2017/Docs/overview.html#benchmarks

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

execution time and performance. Suppose we have two computers, X and Y. When we say, X is n times faster than Y, we mean

$$\frac{\text{Execution time}_Y}{\text{Execution time}_X} = n = \frac{\text{Performance}_X}{\text{Performance}_Y} \tag{1}$$

where performance is defined to be the reciprocal of execution time. Note that execution time has units (seconds), but ratios are dimensionless.

HPs1.8 then define SPECRatio to be an estimate of performance relative to a baseline. Note that when we compare SPECRatios for two machines, A and B, the baseline conveniently drops out. That is,

$$\frac{\text{SPECRatio}_A}{\text{SPECRatio}_B} = \frac{\text{Performance}_A/\text{baseline}}{\text{Performance}_B/\text{baseline}} \approx \frac{\text{Performance}_A}{\text{Performance}_B} \tag{2}$$

HPs1.8 then introduce the GM:

$$GM = \sqrt[\frac{1}{n}]{\prod_{i=1}^{n} sample_i} = exp\left(\frac{1}{n}\sum_{i=1}^{n} log(sample_i)\right) \tag{3}$$

where $sample_i$ is the SPECRatio for program i. In our benchmarks, we use task i instead of program i.

Why use GMs instead of arithmetic means? HPs1.8 observe a problem with units: *Since SPECRatios have no units, comparing SPECRatios arithmetically is meaningless.* For additional motivation, they observe the following two properties of the GM:

1. *The GM of the ratios is the same as the ratio of the GMs.*
2. *The ratio of the GMs is equal to the GM of the performance ratios, which implies that the choice of the reference computer* [baseline] *is irrelevant.*

The first property is a convenient invariant. Because multiplication is associative, we can compute ratios in a number of different ways. The second property says that the choice of the baseline is not that important for many of the comparisons that we want to make. In particular, if we want to choose between buying machine X and machine Y, then we can compare the SPECRatios for both machines. The fact that both of these ratios are defined to be relative to an old VAX is not relevant.

That said, we will suggest below that ratios in our field should be defined in terms of something more meaningful such as human performance. If we did that, then ratios can be interpreted as performance of a system relative to human performance.

HPs1.8 conclude with: *the motivations to use the GM are substantial, especially when we use performance ratios to make comparisons.*

GMs have also been suggested in Information Retrieval (Robertson 2006), though at a more micro level. The discussion above uses GMs to aggregate performance over programs (tasks), whereas Robertson (2006) uses GMs to aggregate over items in a test set. This suggestion runs into difficulties with zeros (Ravana and Moffat 2008). Robertson (2006) introduces an arbitrary small number, ϵ, to smooth GM scores:

$$\begin{aligned} \text{AL}_\epsilon(x_1, \ldots, x_n) &= \frac{1}{n}\sum_{i=1}^{n} log(x_i + \epsilon) \\ \text{GM}_\epsilon(x_1, \ldots, x_n) &= exp(\text{AL}_\epsilon(x_1, \ldots, x_n) - \epsilon \end{aligned} \tag{4}$$

Adding ϵ is easy to implement though there are better ways to smooth small counts (Gale and Church 1994). If possible, it may be wise to avoid methods that depend too much on smoothing of

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

Table 4. SuperGLUE results from Table 9 in (Sun *et al.* 2021). Humans are better than machines on 4 of 8 **tasks**. If the outlier in yellow is dropped, Human is ahead

Model	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	Score
Human	89.0	95.8/98.9	100	81.8/ 51.9	91.7/91.3	93.6	80.0	100	89.8
T5+Menna	**91.4**	95.8/97.6	98.0	88.3/63.0	94.2/93.5	93.0	**77.9**	96.6	90.4
DeBERTa	90.4	95.7/97.6	**98.4**	88.2/**63.7**	94.5/94.1	**93.2**	77.5	95.9	90.3
ERNIE3.0	91.0	**98.6/99.2**	97.4	**88.6**/63.2	**94.7/94.2**	92.6	77.4	97.3	**90.6**

small counts (Good 1953; Goodman 2001). Smoothing is necessary when aggregating over items in a test set, but less so when aggregating over tasks.

9. Mashey's criticisms of benchmarking in NLP

This section is based on Mashey's talk, "Lessons from SPEC," especially the underlined criticism:

I went to SuperGLUE. I looked at the leaderboard, which ended up being a lot like the tables we used to do for SPEC. The one at the top was ERNIE 3.0. I looked at the paper. It was a like a ***1980s performance brief with lots and lots of benchmarks and comparisons with other people when they could find them.***[n]

This criticism is specifically in reference to Table 4, but is also applicable to many of our benchmarks which use arithmetic means to average over a number of tasks:

- GLUE (Wang *et al.* 2018): CoLA, SST-2, MRPC, STS-B, QQP, MNLI, QNLI, RTE, WNLI
- SuperGLUE (Wang *et al.* 2019): BoolQ, CB, COPA, MultiRC, ReCoRD, RTE, WiC, WSC
- MRQA (Fisch *et al.* 2019): SQuAD, NewsQA, TriviaQA, SearchQA, HotpotQA, Natural Questions, BioASQ, DROP, DuoRC, RACE, RelationExtraction, TextbookQA, BioProcess, ComplexWebQ, MCText, QAMR, QAST, TREC
- SciRepEval[o] (Singh *et al.* 2023): biomimicry, cite_count, cite_prediction, cite_prediction_new, drsm, fos, high_influence_cite, mesh_descriptors, nfcorpus, paper_reviewer_matching, peer_review_score_hindex, pub_year, relish, same_author, scidocs_mag_mesh, scidocs_view_cite_read, search, trec_covid, tweet_mentions
- BIG-bench (Srivastava *et al.* 2023): 204 tasks, contributed by 450 authors across 132 institutions.

Mashey's slides[p] raise a few additional criticisms:

1. Incompatible units (slide 22): Does it make sense to average accuracy, precision, F1, correlations, etc.?
2. Headroom (slide 23): GLUE was replaced with SuperGLUE because "progress of the last twelve months has eroded headroom" Mashey points out that this comment is "Reminiscent of SPEC in early days."

[n] https://youtu.be/KmwFw0GHET4?t=1972
[o] https://huggingface.co/datasets/allenai/scirepeval
[p] https://github.com/kwchurch/Benchmarking_past_present_future/blob/master/slides/session3/Mashey2.pptx

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

Table 5. Results from Table 4, relative to human performance. GM and GM_{robust} are geometric means with and without the outlier in yellow

Model	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	GM	GM_{robust}
T5+Menna	1.03	1.00/0.99	0.98	1.08/ 1.21	1.03/1.02	0.99	0.97	0.97	1.02	0.98
DeBERTa	1.02	1.00/0.99	0.98	1.08/ 1.23	1.03/1.03	1.00	0.97	0.96	1.01	0.98
ERNIE 3.0	1.02	1.03/1.00	0.97	1.08/ 1.22	1.03/1.03	0.99	0.97	0.97	1.01	0.98

3. Outliers and Means (slide 25): Problematic usage of arithmetic means. Conclusions depend on an outlier in yellow. Some tasks highlighted in blue point in one direction and other tasks point in the other direction.

Mashey continues by noticing that the standard deviations for humans in Table 4 are much larger than for machines because of an outlier highlighted in yellow. How robust is the conclusion that machines are performing better than humans? He performed a standard robustness test and found the conclusion depends on the outlier. That is, machines are better than people with the outlier, but not without the outlier. We should be very concerned about this lack of robustness. How confident are we in the outlier?

The blue highlighting in Table 4 calls out another concern with robustness that Mashey mentioned. Humans are better than machines on some tasks highlighted in blue, and worse on other tasks. We should be concerned that half of the tasks point in one direction and half point in the other direction. Our field should do more error analysis to understand why the blue tasks are different from the other tasks, as well as why the yellow outlier is so different from all the other tasks.

It is easier to say with confidence that one method is better than another when all/most of the tasks point in the same direction. When there is a split decision, then it is likely *your mileage may vary*. That is, one method might be better for some users and another method might be better for other users. For example, in the case of CPUs, one could imagine that some machines might be better for scientific computing and other machines might be better for memory bound jobs. Consequently, there might be a split decision where some programs perform better on one machine and other programs perform better on another machine. While there may be good reasons for split decisions, split decisions are more complicated than unanimous decisions.

If possible, we should dive deeper into split decisions to understand why they split the way they do. We need to help users make sensible decisions. Which machine/foundation model should they use? The answer becomes complicated if it depends on factors we hope to abstract over such as use cases.

10. Reporting scores relative to human performance

Mashey is also concerned about the use of arithmetic means, which is especially problematic given incompatible units. (SuperGLUE averages scores on different scales such as accuracy and F1.) Table 5 addresses this concern. By reporting results relative to human performance, all the scores are dimensionless ratios. In addition, by reporting scores as GMs, then all of the ratios can be interpreted as performance relative to human performance.

Unfortunately, Mashey's concern about the outlier remains an issue in Table 5. Note that the scores in the GM column are above 1 and the scores in the GM_{robust} column are below 1, indicating that machines are better than humans with the outlier, but not if the outlier is removed. The blue highlighting calls out the observation that humans are better on half of the tasks. This is even easier to see in Table 5 where the blue columns have ratios less than 1, whereas the other columns

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

have ratios greater than 1. We need to understand why some tasks appear to be easier for machines and others appear to be easier for humans. There used to be more of this kind of error analysis in ACL papers. We need more deep-dives and exploratory data analysis (Hoaglin *et al.* 2000; Hoaglin 2003).

11. Additional concerns with benchmarks in our field

There are a few additional concerns:

1. Simplicity
2. Validity and reliability
3. Maintenance and processes for updating benchmarks
4. Challenging, Meaningful, and Informative tasks

11.1 Simplicity

Benchmarks should be as simple as possible, and no simpler. As mentioned above, "benchmarks based on running programs that are much simpler than a real application have led to performance pitfalls." But on the other hand, many of our recent benchmarks are becoming larger and larger over time. Newer benchmarks such as BIG-bench are often created by combining more and more tasks, often by combining older benchmarks. When benchmarks become more complicated than necessary, it becomes too difficult to perform error analysis.

11.2 Validity and reliability

Validity and reliability are discussed in Krippendorff (2018). Reliability is about data, and validity is about truth (and use cases).

1. Reliability: Are results repeatable? Do we trust the gold labels? Is there reasonable inter-annotator agreement?
2. Validity: How confident are we that these tasks span the space that needs to be covered for evaluating general purpose foundation models?

11.3 Maintenance and processes for updating benchmarks

There are organizations such as SPEC and ML Commons with a mandate to keep their benchmarks up to date, but most benchmarks in our field lack this type of support. It is inevitable that new tasks will need to be added as technology evolves. So too, there need to be processes for retiring old tasks, as tasks become less challenging, meaningful, or informative. We also need processes to calibrate scores computed over different sets of tasks in order to establish improvements over years/decades.

11.4 Challenging, meaningful, and informative tasks

Tasks should be informative. Tasks should be retired when the community is spending too much time on pointless exercises that do not generalize well to real workloads from real customers

In addition, we will not learn much from tasks that are too easy or too hard. Consider the Winograd Schema, a task in GLUE. The system is given input sentences such as

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

- The city councilmen refused the demonstrators a permit because they *feared* violence.
- The city councilmen refused the demonstrators a permit because they *advocated* violence.

Depending on the underlined word, *they* refers to *city councilmen* or *demonstrators*. The task is often formulated as binary classification.

See Kocijan *et al.* (2023) for the history of the Winograd Schema. This task was long considered "AI-Complete." The main point of Kocijan *et al.* (2023), though, is that systems have recently "defeated" the Winograd task. That is, systems have found a way to perform considerably better than baseline, though not in interesting ways that are likely to generalize well to real workloads from real customers. Tasks such as this should be retired from benchmarks when they have been "defeated."

12. Principles for measuring and reporting performance

Since it takes time for the community to develop and implement great benchmarks, it is better to start small and focus on common use cases first to debug the methodology. By adopting some of the SPEC techniques, we can create and use benchmarks that are more robust, efficient, reliable, and credible. The rest of this paper outlines principles that we think are appropriate for the NLP community.

12.1 Interpretable scores

As mentioned above, SPECRatios can be interpreted as multiplicative performance gains relative to as baseline (VAX 11/780 in 1978). Because of their use of GMs, SPECRatios can be used to compare two more recent machines relative to one another. The details of the baseline drop out when making these comparisons.

We suggest that our scores should also be defined as ratios, where the baseline would be human performance. This implies that we replace arithmetic means with geometric means when aggregating scores (ratios) over tasks.

12.2 Focus on real use cases that are challenging and informative

When measuring CPU performance, HPs1.8 suggest we focus on real applications and avoid synthetic (or artificially constructed) programs. Some papers in our field refer to tasks in popular benchmarks as "real," but many of them are more synthetic than real. We have little reason to believe that these tasks are representative of customer experience, or that they span the space of common use cases.

Too many of the tasks in our benchmarks encourage pointless SOTA-chasing in ways that do not generalize to customer experience. SPEC removed tasks that were insufficiently informative because they were too easy or too hard, or because they encouraged pointless SOTA-chasing.

As technology and customer expectations evolve over time, benchmarks need to be updated. If a task is no longer relevant (or challenging or informative), it should be retired. This is easier to say than done, but it will not happen unless we make it a priority.

12.3 Easy to reproduce

The guiding principle of reporting performance is reproducibility. That is, all the details that are needed so another person can replicate the results. This includes configurations and other specific information like prompts to avoid any confusion. The amount of development required to reproduce results should be minimal.

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

12.4 Rigorous development process

Care should be taken when implementing a benchmark. Every program requires a verification test to assure correct execution and output metrics. Clearly document system configurations, datasets, prompts, libraries, etc.; transparent reporting of metrics is a must. Best practices suggest using statistical methods to analyze distributions. Constantly look for suspicious outliers. Allocate time for error analysis.

13. Conclusions

There are many lessons to be learned from SPEC. SPEC has had more influence than benchmarks in our field. By addressing a simple commercially important question, which computer should I buy, SPEC helped the computer industry to become as important as it has become. As mentioned above, SPEC made a convincing credible quantitative statement about computer speeds over decades. It is remarkable that we can make statements that generalize to what customers are likely to expect even though different customers will use these machines to do different things.

We view foundation models as similar to general purpose computers, where fine-tuning and prompt-engineering is analogous to programming an Intel 4004 microprocessor. Evaluating general purpose solutions is different from evaluating special purpose solutions. The challenge is to find ways to evaluate foundation models so we can make quantitative statements about improvements of general purpose foundation models over years/decades without specifying tasks, datasets, prompts, etc. In section 10, as a small step in this direction, we suggested normalizing scores relative to human performance, and replacing arithmetic means with GMs.

From this perspective, we view SPEC as an existence proof that it is possible to make statements about general purpose solutions that generalize over details (programs, inputs, etc.). If we succeed, then there is hope that the entire LLM eco-system could benefit over the next few decades in ways the resemble the growth of the tech sector over the past few decades. By the LLM eco-system, we are referring not only to the companies that build foundation models but also the much larger community that interacts with such models both directly and indirectly.

The definition of performance depends on what matters. In the case of SPEC, performance was defined in terms of execution time whereas for foundation models, performance is defined in terms of other metrics such as accuracy, F1, correlations, etc.

There are clearly opportunities for simplify our benchmarks. We have too many metrics and too many tasks. One of the reasons that SPEC has been as successful as it has been is simplicity.

SPEC has also benefited from experience. Mashey criticized one of our papers as a typical *1980s performance brief with lots and lots of benchmarks and comparisons with other people when they could find them*. The implication is that the state of evaluation in our field is similar to where they were before the tech-savvy bar-owner challenged them to work together to improve the state of the art in evaluation of general purpose computing. It will take time for benchmarking in our field to learn from their experience and develop effective evaluations of general purpose foundation models.

Another huge advantage of SPEC is cooperation (coopetition). Ellen Voorhees, an organizer of many TREC events, is a strong advocate of coopetition, as discussed in her contribution to the ACL-2021 workshop on benchmarking.[q] Mashey also discusses coopetition in his "Lessons from SPEC."[r] In response to a question from the press:

These are done by computer companies. Isn't that letting foxes guard the henhouse?

[q] https://github.com/kwchurch/Benchmarking_past_present_future?tab=readme-ov-file#Voorhees
[r] Slide 14 of footnote [p]

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

Mashey responded with:

Nobody is better than a fox at keeping other foxes from eating the hens.

Mashey continued with:

Coopetition of fierce competitors can actually work well, once culture of trust and honesty built, and people find benefits from avoiding duplicative efforts.

In short, we need an organization like SPEC or ML Commons to maintain benchmarks so they challenge the community to make continuous progress and avoid pointless SOTA-chasing.

References

Broder A. (2002). A taxonomy of web search. *ACM SIGIR Forum* **36**(2), 3–10.

Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I. and Amodei D. (2020). Language models are few-shot learners. In Larochelle H, Ranzato M, Hadsell R, Balcan MF and Lin H. (eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc., pp. 1877–1901.

Calzarossa M. C., Massari L. and Tessera D. (2016). Workload characterization: A survey revisited. *ACM Computing Surveys (CSUR)* **48**(3), 1–43.

Chowdhery A., Narang S., Devlin J., Bosma M., Mishra G., Roberts A., Barham P., Chung H. W., Sutton C., Gehrmann S. (2023). PaLM: scaling language modeling with pathways. *Journal of Machine Learning Research* **24**(240), 1–113

Church K., Liberman M. and Kordoni V. (2021). Benchmarking: Past, present and future. In Church, K., Liberman, M., and Kordoni, V. (eds), *Proceedings of the 1st Workshop on Benchmarking: Past, Present and Future*, Online. Association for Computational Linguistics, pp. 1–7.

Church K. W. (2022). Emerging trends: deep nets thrive on scale. *Natural Language Engineering* **28**(5), 673–682.

Church K. W. and Kordoni V. (2022). Emerging trends: SOTA-chasing. *Natural Language Engineering* **28**(2), 249–269.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding, In Burstein, J., Doran, C., and Solorio, T. (eds), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol 1 (Long and Short Papers), Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186.

Dixit K. M. (1993). Overview of the SPEC benchmarks. *The Benchmark Handbook* **7**.

Faggin F. (2018). How we made the microprocessor. *Nature Electronics* **1**(1), 88–88.

Fisch A., Talmor A., Jia R., Seo M., Choi E. and Chen D. (2019). MRQA. 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, ACL, pp. 1–13.

Gale W. A. and Church K. W. (1994). What's wrong with adding one. In *Corpus-Based Research into Language: In Honour of Jan Aarts*, pp. 189–200.

Giladi R. and Ahituv N. (1995). SPEC as a performance evaluation measure. *Computer* **28**(8), 33–42.

Good I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika* **40**(3–4), 237–264.

Goodman J. (2001). A bit of progress in language modeling. ArXiv, cs.CL/0108005

He K., Zhang X., Ren S. and Sun J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778

Hennessy J. L. and Patterson D. A. (2012). *Computer Architecture: A Quantitative Approach*. Elsevier.

Hoaglin D. C. (2003). John W. Tukey and data analysis. *Statistical Science* **18**(3), 311–318.

Hoaglin D. C., Mosteller F. and Tukey J. W. (2000). *Understanding Robust and Exploratory Data Analysis*, vol. **76**. John Wiley & Sons.

Kocijan V., Davis E., Lukasiewicz T., Marcus G. and Morgenstern L. (2023). *The Defeat of the Winograd Schema Challenge. Artificial Intelligence* 325, 103971. https://doi.org/10.1016/j.artint.2023.103971.

Krippendorff K. (2018). *Content Analysis: An Introduction to Its Methodology*. London: Sage publications.

Mashey J. (2004). War of the benchmark means: time for a truce. *SIGARCH Comput. Archit. News* **32**(4), 1–14.

Mashey J. (2005). Summarizing performance is no mean feat [computer performance analysis]. In *IEEE International. 2005 Proceedings of the IEEE Workload Characterization Symposium 2005*, pp. 1.

Radford A., Wu J., Child R., Luan D., Amodei D. and Sutskever I. (2019). *Language Models Are Unsupervised Multitask Learners*. OpenAI Blog.

https://doi.org/10.1017/S1351324924000068 Published online by Cambridge University Press

Ravana S. D. and Moffat A. (2008). Exploring evaluation metrics: GMAP versus MAP. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY. https://doi.org/10.1145/1390334.1390452.

Robertson S. (2006). On GMAP: and other transformations. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 78–83.

Singh A., **D'Arcy M.**, **Cohan A.**, **Downey D. and Feldman S.** (2023). SciRepEval: A multi-format benchmark for scientific document representations. In Bouamor, H., Pino, J., and Bali, K. (eds), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore: Association for Computational Linguistics, pp. 5548–5566.

Srivastava A., **Rastogi A.**, **Rao A.**, **Shoeb A. A. M.**, **Abid A.**, **Fisch A.**, **Brown A. R.**, **Santoro A.**, **Gupta A.**, **Garriga-Alonso A.**, **Kluska A.**, **Lewkowycz A.**, **Agarwal A.**, **Power A.**, **Ray A.**, **Warstadt A.**, **Kocurek A. W.**, **Safaya A.**, **Tazarv A.**, . . . **Wu Z.** (2023). Beyond the imitation game: quantifying and extrapolating the capabilities of language models. *Transactions On Machine Learning Research*

Sun Y., **Wang S.**, **Feng S.**, **Ding S.**, **Pang C.**, **Shang J.**, **Liu J.**, **Chen X.**, **Zhao Y.**, **Lu Y. and et al.** (2021). Ernie 3.0: large-scale knowledge enhanced pre-training for language understanding and generation, arXiv preprint arXiv: 2107.0.2137.

Turing A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Journal of Mathematics* **58**(345-363), 5.

Wang A., **Pruksachatkun Y.**, **Nangia N.**, **Singh A.**, **Michael J.**, **Hill F.**, **Levy O. and Bowman S.** (2019). SuperGLUE: a stickier benchmark for general-purpose language understanding systems. In Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., and Garnett, R., (eds), *Advances in Neural Information Processing Systems*, vol. **32**, Curran Associates, Inc.

Wang A., **Singh A.**, **Michael J.**, **Hill F.**, **Levy O. and Bowman S.** (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Association for Computational Linguistics, Brussels, Belgium, pp. 353–355. https://doi.org/10.18653/v1/W18-5446

Zhang M., **Martin P.**, **Powley W. and Chen J.** (2018). Workload management in database management systems: a taxonomy. *IEEE Transactions On Knowledge and Data Engineering* **30**(7), 1386–1402.

Cite this article: Church KW and Alonso O (2024). Emerging trends: evaluating general purpose foundation models. *Natural Language Engineering* **30**, 1323–1335. https://doi.org/10.1017/S1351324924000068