Questionable Answers in Question Answering

Matt Crane

> “Experiments vary greatly in goal and scale, but always rely on repeatable procedure and logical analysis of the results.”
>
> – Wikipedia: Experiment

“Based on theoretical reasoning it has been suggested that the reliability of findings published in the scientific literature decreases with the popularity of the research field.”

– Pfeiffer and Hoffmann, 2009

> “Based on theoretical reasoning it has been suggested that the reliability of findings published in the scientific literature decreases with the popularity of the research field.”
>
> – Pfeiffer and Hoffmann, 2009

Thomas Pfeiffer and Robert Hoffmann. 2009. Large-scale assessment of the effect of popularity on the reliability of research. *PLoS One*, 4(6):e5996.

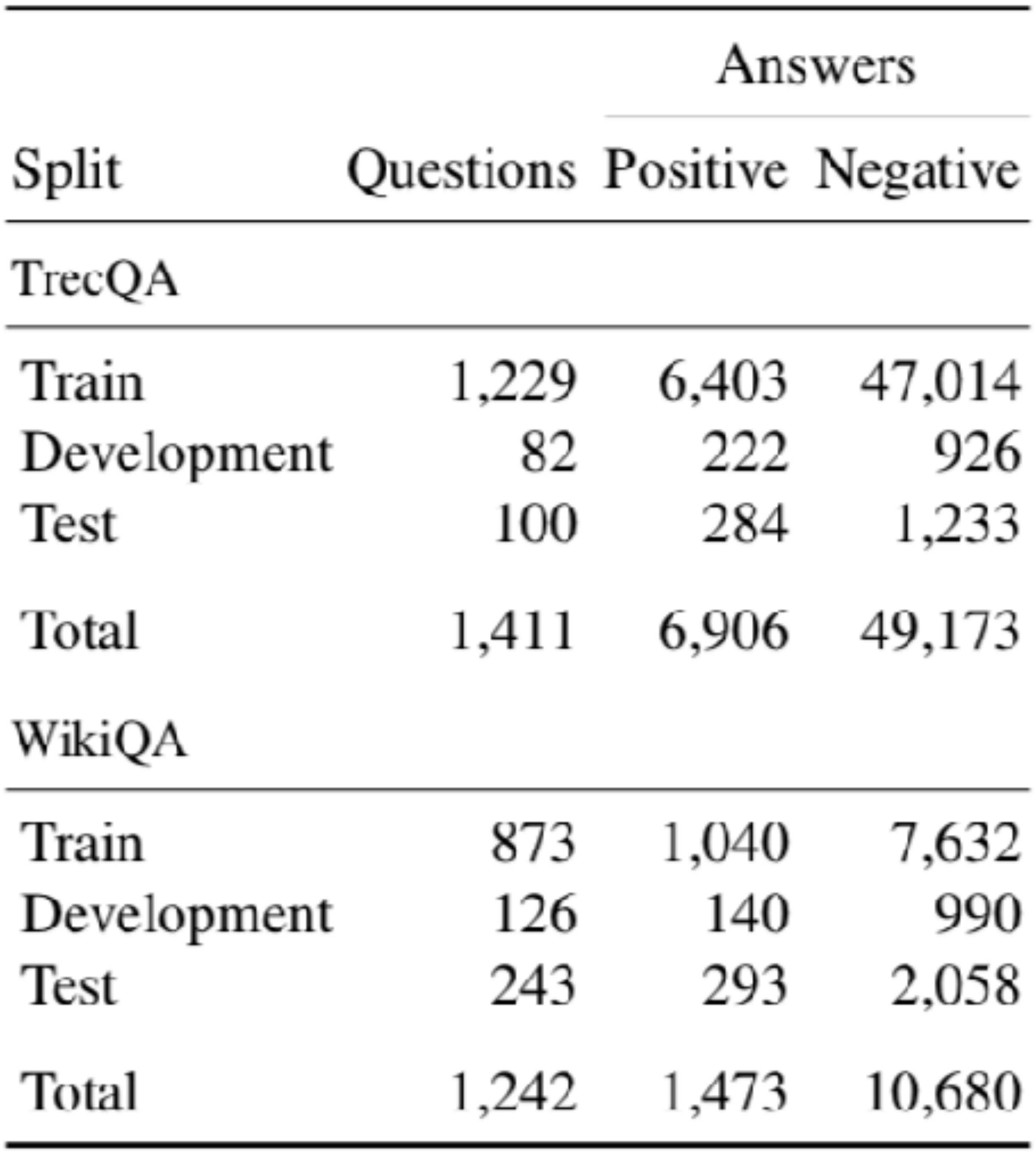
Introduction

- Reproducibility and replicability are fundamental aspects of science
- Deep-learning is a very popular field
- There are a number of unreported environmental elements that produce variation that is at least as much as reported improvements.

Exemplar Task/Model

- Question answering over free text: given a question and a set of candidate sentences, rank those sentences based on likelihood that the sentence contains an answer to the question
  - Example question: what was the monetary value of the nobel peace prize in 1989 ?
  - Example candidate sentence: each nobel prize is worth $ 469,000 .
- Example model is an implementation of the SM model, previously described by members of this lab in prior presentations/seminars

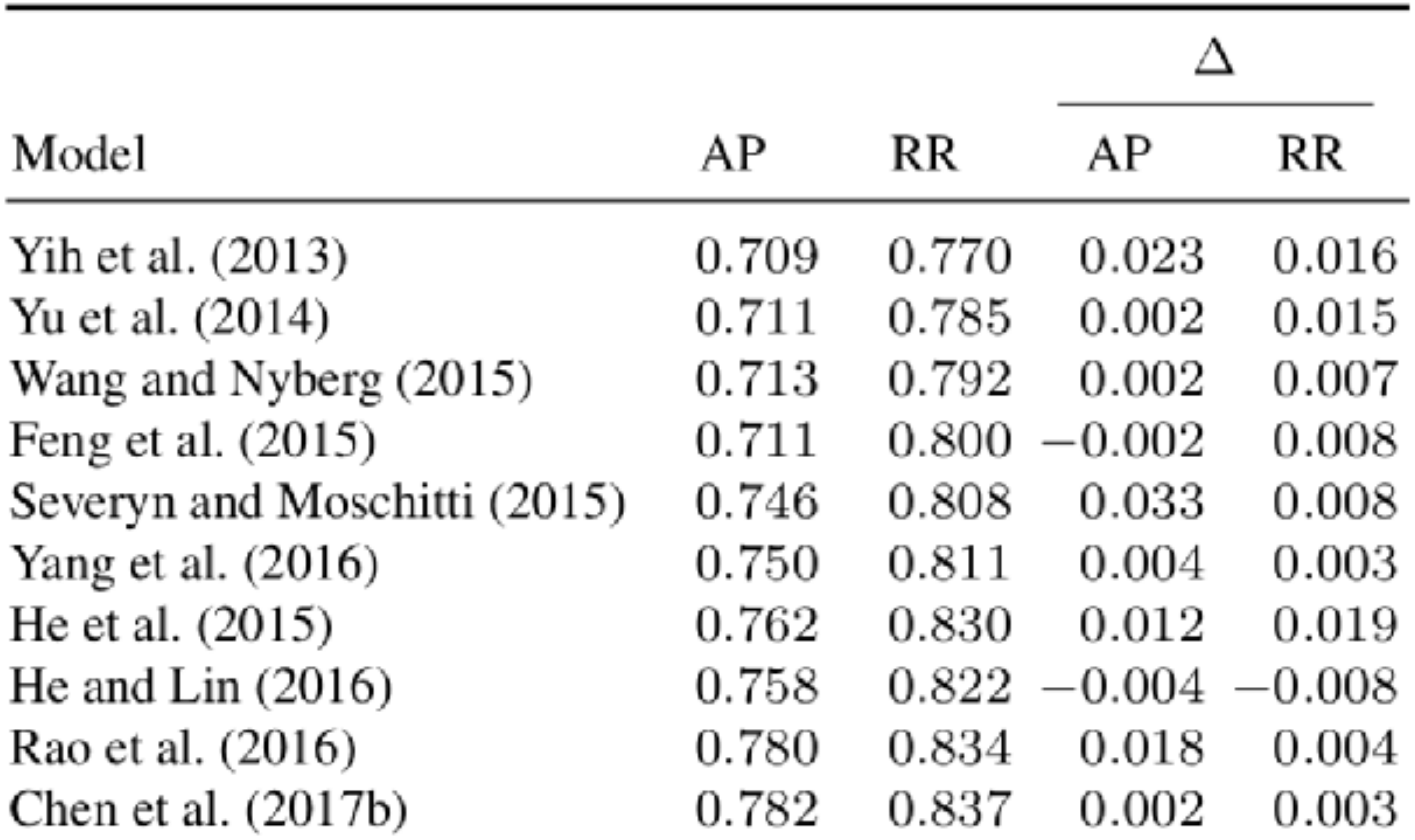
Datasets



		Answers	
Split	Questions	Positive	Negative
TrecQA			
Train	1,229	6,403	47,014
Development	82	222	926
Test	100	284	1,233
Total	1,411	6,906	49,173
WikiQA			
Train	873	1,040	7,632
Development	126	140	990
Test	243	293	2,058
Total	1,242	1,473	10,680

Progress

- Some of the later results from the TrecQA dataset:

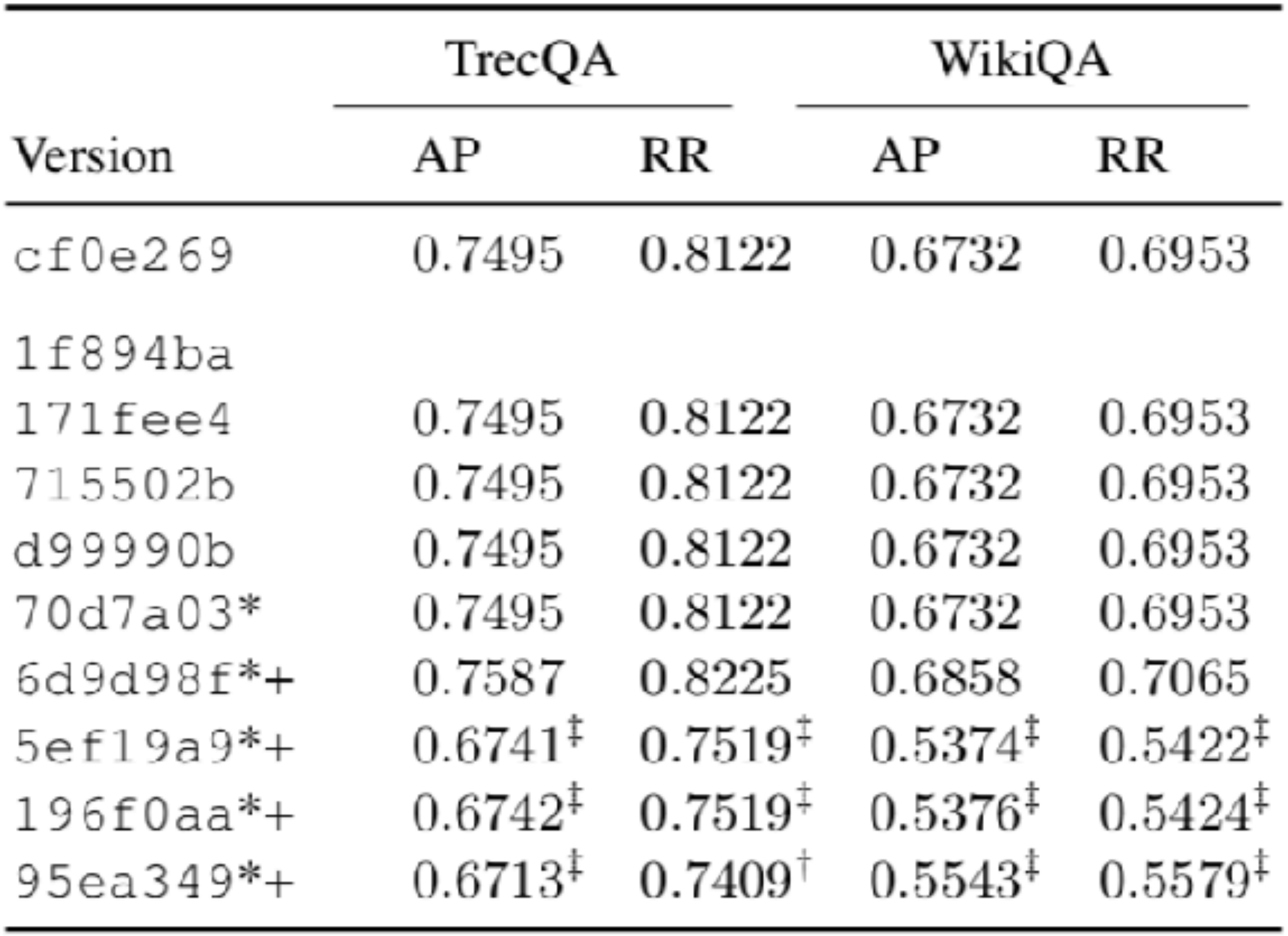


			Δ	
Model	AP	RR	AP	RR
Yih et al. (2013)	0.709	0.770	0.023	0.016
Yu et al. (2014)	0.711	0.785	0.002	0.015
Wang and Nyberg (2015)	0.713	0.792	0.002	0.007
Feng et al. (2015)	0.711	0.800	−0.002	0.008
Severyn and Moschitti (2015)	0.746	0.808	0.033	0.008
Yang et al. (2016)	0.750	0.811	0.004	0.003
He et al. (2015)	0.762	0.830	0.012	0.019
He and Lin (2016)	0.758	0.822	−0.004	−0.008
Rao et al. (2016)	0.780	0.834	0.018	0.004
Chen et al. (2017b)	0.782	0.837	0.002	0.003

- Note the small changes in AP/RR

Software Versions

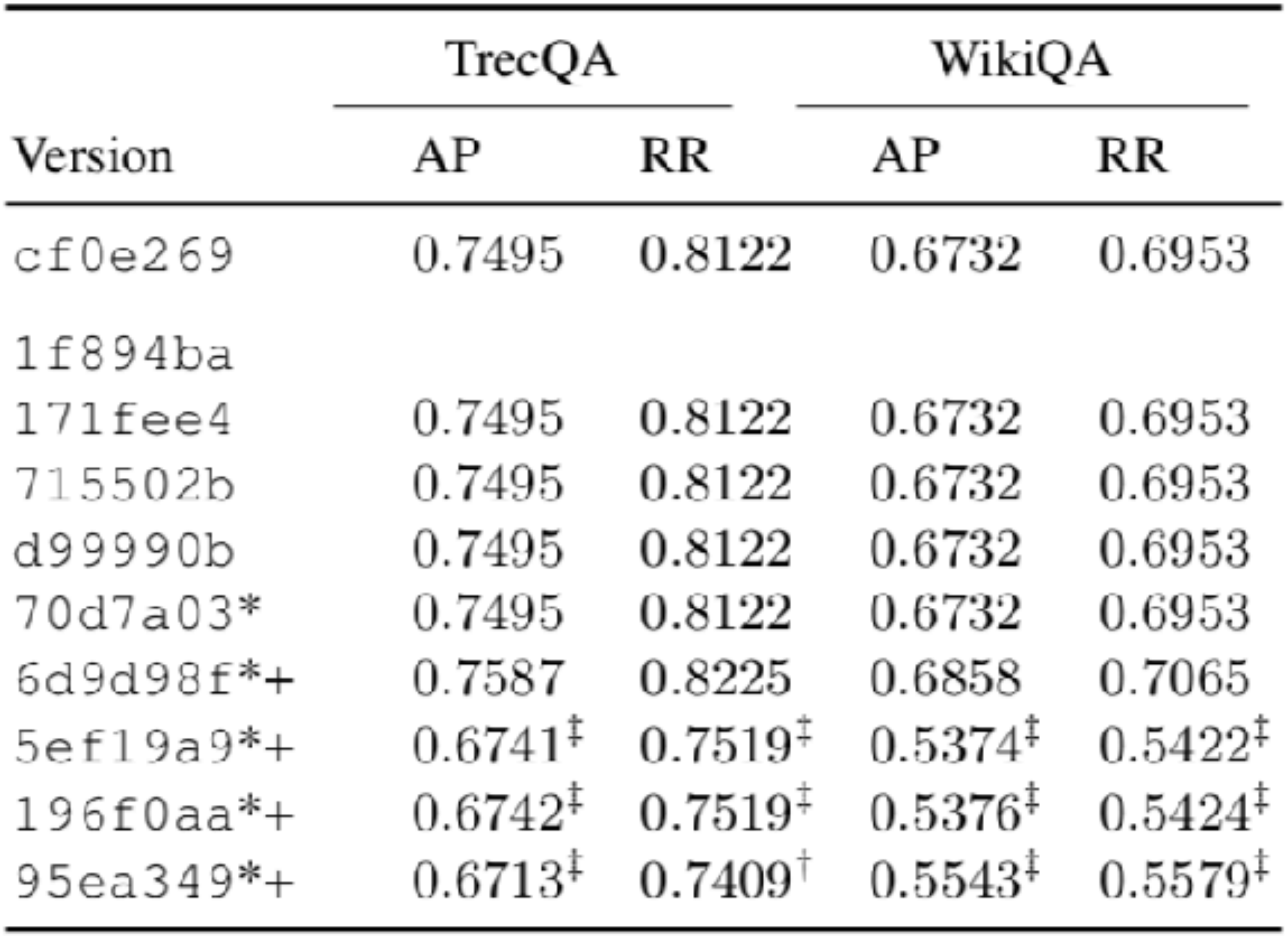
- Nobody writes perfect code, and when we change the code, we change the results…



	TrecQA		WikiQA	
Version	AP	RR	AP	RR
cf0e269	0.7495	0.8122	0.6732	0.6953
1f894ba				
171fee4	0.7495	0.8122	0.6732	0.6953
715502b	0.7495	0.8122	0.6732	0.6953
d99990b	0.7495	0.8122	0.6732	0.6953
70d7a03*	0.7495	0.8122	0.6732	0.6953
6d9d98f*+	0.7587	0.8225	0.6858	0.7065
5ef19a9*+	0.6741‡	0.7519‡	0.5374‡	0.5422‡
196f0aa*+	0.6742‡	0.7519‡	0.5376‡	0.5424‡
95ea349*+	0.6713‡	0.7409†	0.5543‡	0.5579‡

Software Versions

- Nobody writes perfect code, and when we change the code, we change the results…

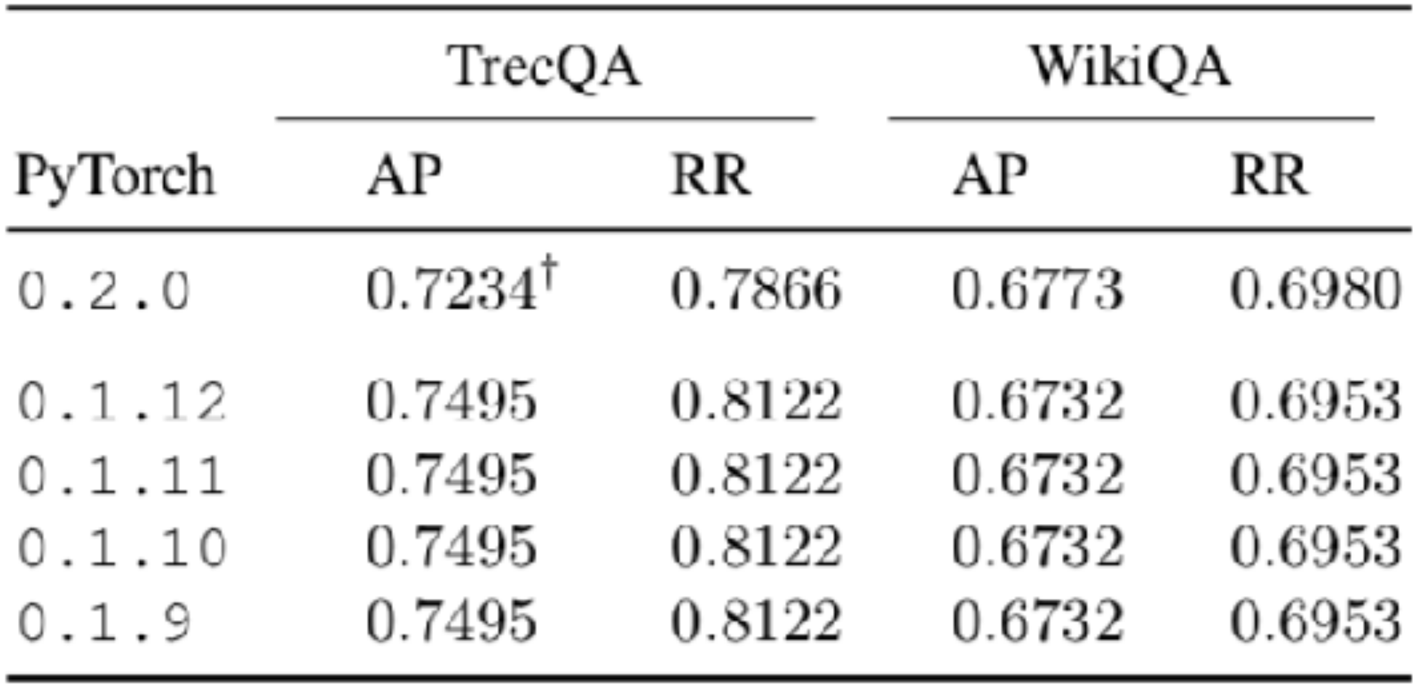


	TrecQA		WikiQA	
Version	AP	RR	AP	RR
`cf0e269`	0.7495	0.8122	0.6732	0.6953
`1f894ba`				
`171fee4`	0.7495	0.8122	0.6732	0.6953
`715502b`	0.7495	0.8122	0.6732	0.6953
`d99990b`	0.7495	0.8122	0.6732	0.6953
`70d7a03*`	0.7495	0.8122	0.6732	0.6953
`6d9d98f*+`	0.7587	0.8225	0.6858	0.7065
`5ef19a9*+`	$0.6741^{\ddagger}$	$0.7519^{\ddagger}$	$0.5374^{\ddagger}$	$0.5422^{\ddagger}$
`196f0aa*+`	$0.6742^{\ddagger}$	$0.7519^{\ddagger}$	$0.5376^{\ddagger}$	$0.5424^{\ddagger}$
`95ea349*+`	$0.6713^{\ddagger}$	$0.7409^{\dagger}$	$0.5543^{\ddagger}$	$0.5579^{\ddagger}$

- … significantly ($p < 0.01^{\ddagger}$, $p < 0.05^{\dagger}$ against `cf0e269`, paired Wilcoxon signed rank test)

Framework Versions

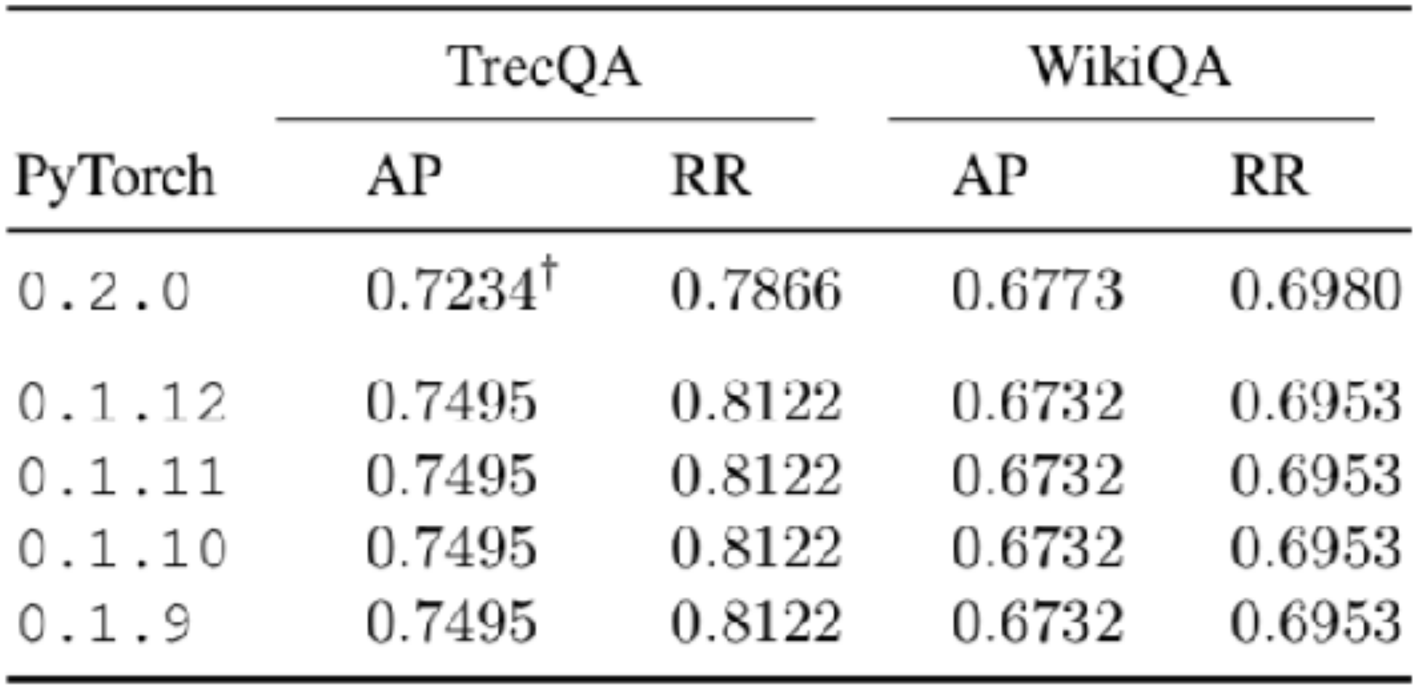
- Sometimes the framework you use makes changes, sometimes to the bits of the framework that you use…



	TrecQA		WikiQA	
PyTorch	AP	RR	AP	RR
0.2.0	$0.7234^{\dagger}$	0.7866	0.6773	0.6980
0.1.12	0.7495	0.8122	0.6732	0.6953
0.1.11	0.7495	0.8122	0.6732	0.6953
0.1.10	0.7495	0.8122	0.6732	0.6953
0.1.9	0.7495	0.8122	0.6732	0.6953

Framework Versions

- Sometimes the framework you use makes changes, sometimes to the bits of the framework that you use…



	TrecQA		WikiQA	
PyTorch	AP	RR	AP	RR
`0.2.0`	0.7234†	0.7866	0.6773	0.6980
`0.1.12`	0.7495	0.8122	0.6732	0.6953
`0.1.11`	0.7495	0.8122	0.6732	0.6953
`0.1.10`	0.7495	0.8122	0.6732	0.6953
`0.1.9`	0.7495	0.8122	0.6732	0.6953

- … significantly ($p < 0.05^\dagger$ against `0.1.12`, paired Wilcoxon signed rank test)

Docker to the rescue!

- Docker is a containerization tool

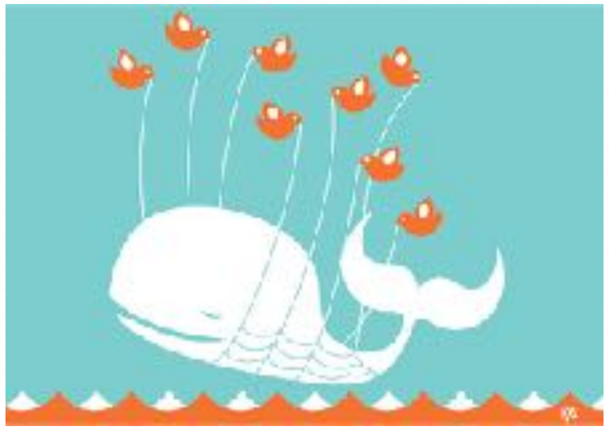
- A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings

- Virtual machines are to hardware what containers are to the operating system

Not Quite

- Still got different answers on different machines, the machines:

- Intel i7-6800K (6 cores, 12 threads)
- AMD FX-8370E (8 cores, 8 threads)
- ‘Intel Xeon’-like on EC2 (2 vCPUs)



Quick Test

- `0.1 + 0.1 + 0.1 == 0.3`

- `0.1 + 0.1 + 0.1 + 0.1 == 0.4`

- `(0.1 + 0.2) + 0.3 == 0.1 + (0.2 + 0.3)`

Quick Test

- `0.1 + 0.1 + 0.1 == 0.3`

  ```
  False
  ```

- `0.1 + 0.1 + 0.1 + 0.1 == 0.4`

- `(0.1 + 0.2) + 0.3 == 0.1 + (0.2 + 0.3)`

Quick Test

-
  ```
  0.1 + 0.1 + 0.1 == 0.3
  ```

  ```
  False
  ```

-
  ```
  0.1 + 0.1 + 0.1 + 0.1 == 0.4
  ```

  ```
  True
  ```

-
  ```
  (0.1 + 0.2) + 0.3 == 0.1 + (0.2 + 0.3)
  ```

Quick Test

- `0.1 + 0.1 + 0.1 == 0.3`

  `False`

- `0.1 + 0.1 + 0.1 + 0.1 == 0.4`

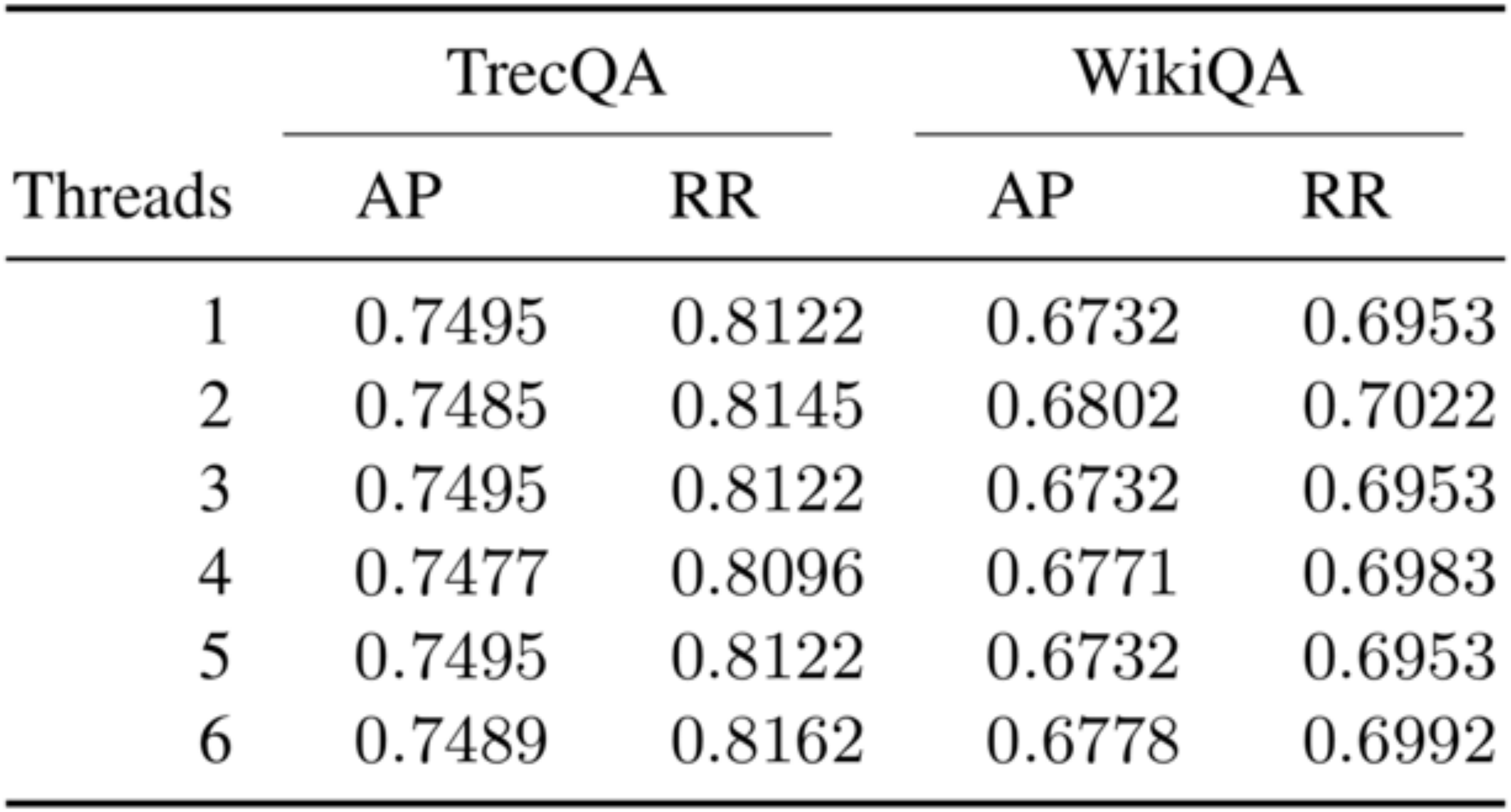
  `True`

- `(0.1 + 0.2) + 0.3 == 0.1 + (0.2 + 0.3)`

  `False`

Threads

- Because of those examples, different numbers of threads give different results, but not because of ordering, but because of workload splitting



	TrecQA		WikiQA	
Threads	AP	RR	AP	RR
1	0.7495	0.8122	0.6732	0.6953
2	0.7485	0.8145	0.6802	0.7022
3	0.7495	0.8122	0.6732	0.6953
4	0.7477	0.8096	0.6771	0.6983
5	0.7495	0.8122	0.6732	0.6953
6	0.7489	0.8162	0.6778	0.6992

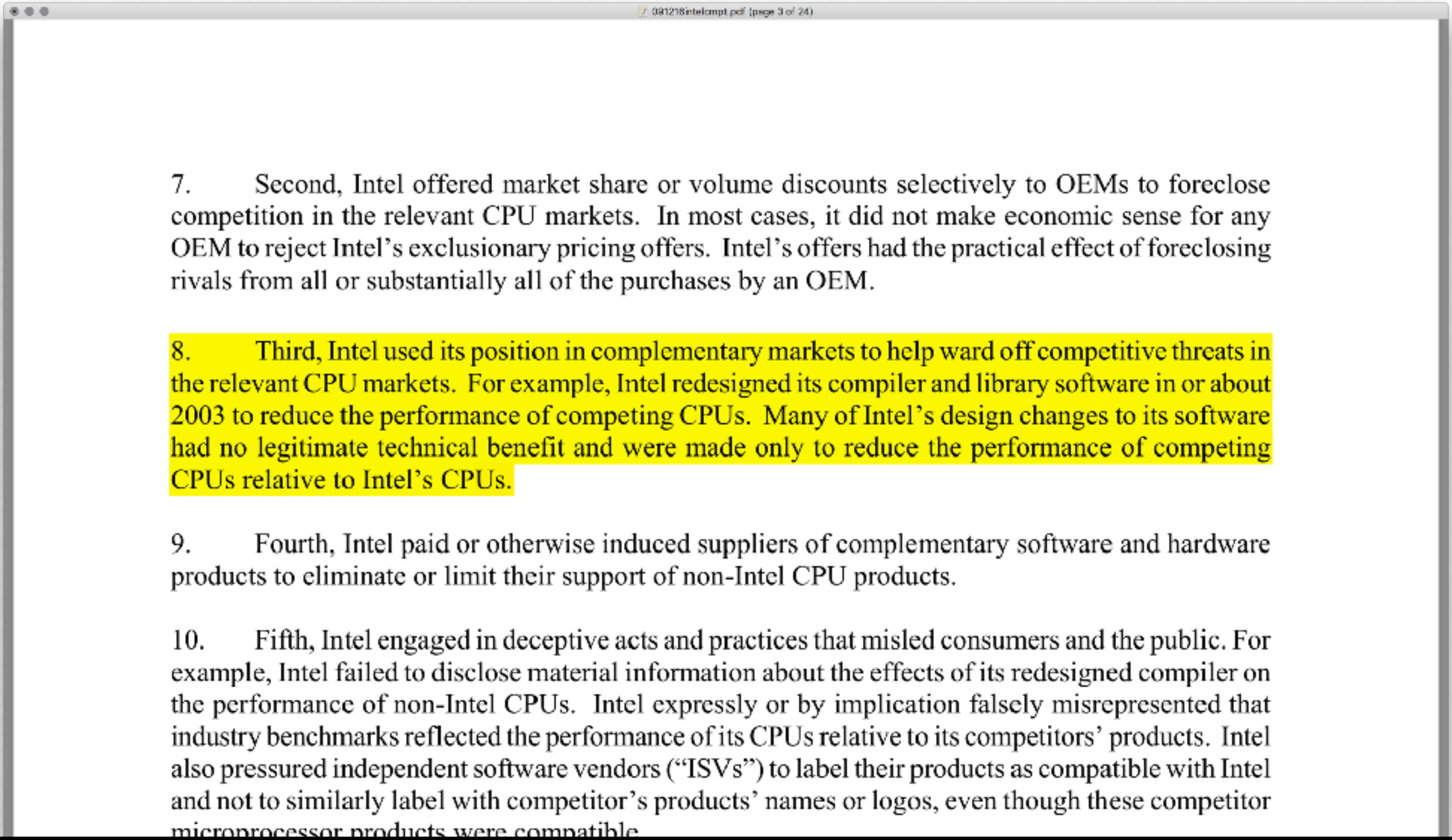
- After fixing threads, now down to two answers

Hardware Differences?

- Intel gives one set of answers, AMD gives another
- Is it possible that different hardware implements the floating point specification differently?
- Yes, but very unlikely given the standard
- Mmm, PyTorch ships with, and uses, the Math Kernel Library by default
    - Written by Intel

It Wouldn’t Be The First Time!

It Wouldn’t Be The First Time!



7. Second, Intel offered market share or volume discounts selectively to OEMs to foreclose competition in the relevant CPU markets. In most cases, it did not make economic sense for any OEM to reject Intel’s exclusionary pricing offers. Intel’s offers had the practical effect of foreclosing rivals from all or substantially all of the purchases by an OEM.

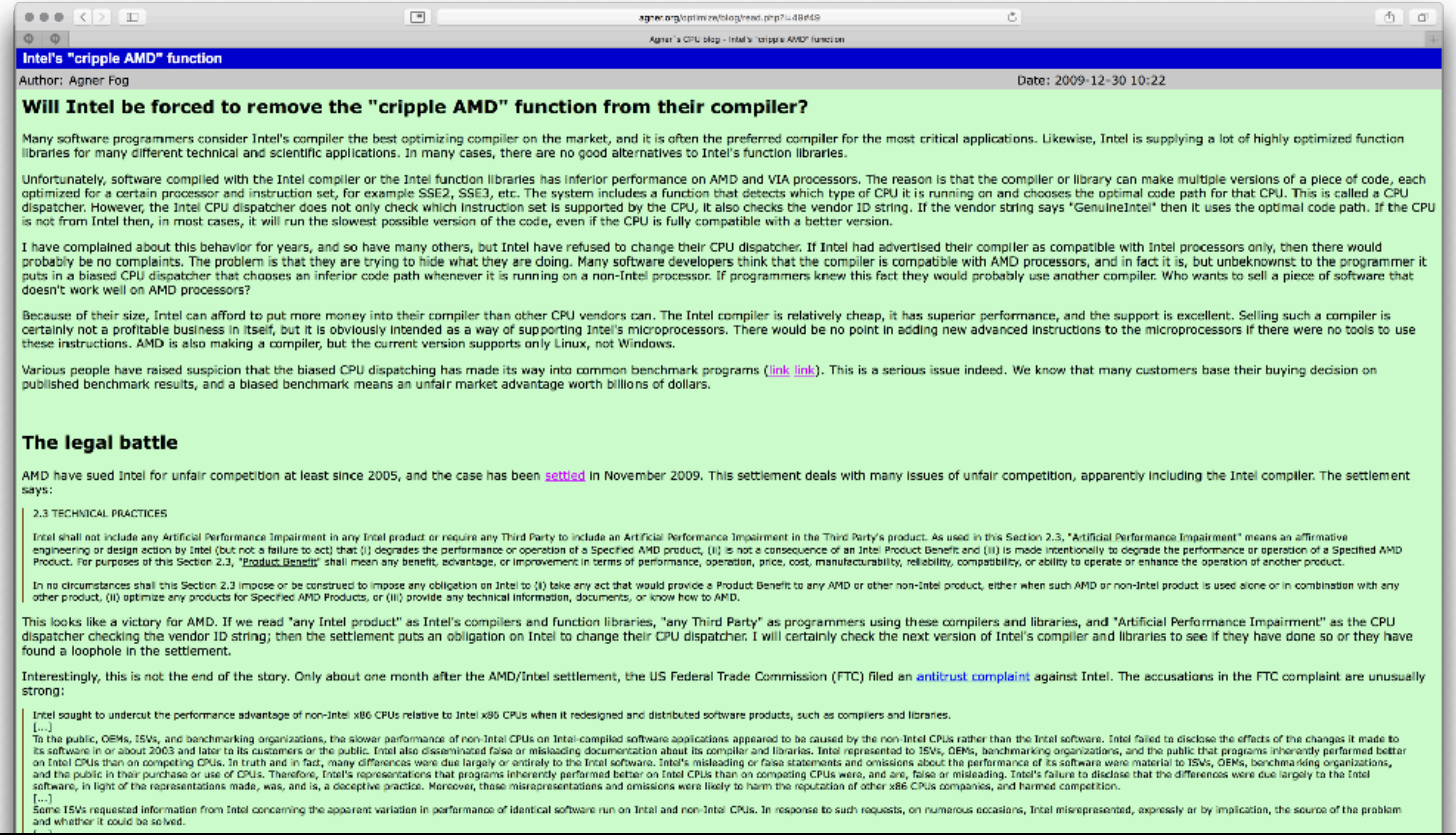
8. Third, Intel used its position in complementary markets to help ward off competitive threats in the relevant CPU markets. For example, Intel redesigned its compiler and library software in or about 2003 to reduce the performance of competing CPUs. Many of Intel’s design changes to its software had no legitimate technical benefit and were made only to reduce the performance of competing CPUs relative to Intel’s CPUs.

9. Fourth, Intel paid or otherwise induced suppliers of complementary software and hardware products to eliminate or limit their support of non-Intel CPU products.

10. Fifth, Intel engaged in deceptive acts and practices that misled consumers and the public. For example, Intel failed to disclose material information about the effects of its redesigned compiler on the performance of non-Intel CPUs. Intel expressly or by implication falsely misrepresented that industry benchmarks reflected the performance of its CPUs relative to its competitors’ products. Intel also pressured independent software vendors (“ISVs”) to label their products as compatible with Intel and not to similarly label with competitor’s products’ names or logos, even though these competitor microprocessor products were compatible.

https://www.ftc.gov/sites/default/files/documents/cases/091216intelcmpt.pdf

It Wouldn’t Be The First Time!


agner.org/optimize/blog/read.php?i=49#49
Agner`s CPU blog - Intel's "cripple AMD" function

Intel's "cripple AMD" function

Author: Agner Fog

Date: 2009-12-30 10:22

Will Intel be forced to remove the "cripple AMD" function from their compiler?

Many software programmers consider Intel's compiler the best optimizing compiler on the market, and it is often the preferred compiler for the most critical applications. Likewise, Intel is supplying a lot of highly optimized function libraries for many different technical and scientific applications. In many cases, there are no good alternatives to Intel's function libraries.

Unfortunately, software compiled with the Intel compiler or the Intel function libraries has inferior performance on AMD and VIA processors. The reason is that the compiler or library can make multiple versions of a piece of code, each optimized for a certain processor and instruction set, for example SSE2, SSE3, etc. The system includes a function that detects which type of CPU it is running on and chooses the optimal code path for that CPU. This is called a CPU dispatcher. However, the Intel CPU dispatcher does not only check which instruction set is supported by the CPU, it also checks the vendor ID string. If the vendor string says "GenuineIntel" then it uses the optimal code path. If the CPU is not from Intel then, in most cases, it will run the slowest possible version of the code, even if the CPU is fully compatible with a better version.

I have complained about this behavior for years, and so have many others, but Intel have refused to change their CPU dispatcher. If Intel had advertised their compiler as compatible with Intel processors only, then there would probably be no complaints. The problem is that they are trying to hide what they are doing. Many software developers think that the compiler is compatible with AMD processors, and in fact it is, but unbeknownst to the programmer it puts in a biased CPU dispatcher that chooses an inferior code path whenever it is running on a non-Intel processor. If programmers knew this fact they would probably use another compiler. Who wants to sell a piece of software that doesn't work well on AMD processors?

Because of their size, Intel can afford to put more money into their compiler than other CPU vendors can. The Intel compiler is relatively cheap, it has superior performance, and the support is excellent. Selling such a compiler is certainly not a profitable business in itself, but it is obviously intended as a way of supporting Intel's microprocessors. There would be no point in adding new advanced instructions to the microprocessors if there were no tools to use these instructions. AMD is also making a compiler, but the current version supports only Linux, not Windows.

Various people have raised suspicion that the biased CPU dispatching has made its way into common benchmark programs (link link). This is a serious issue indeed. We know that many customers base their buying decision on published benchmark results, and a biased benchmark means an unfair market advantage worth billions of dollars.

The legal battle

AMD have sued Intel for unfair competition at least since 2005, and the case has been settled in November 2009. This settlement deals with many issues of unfair competition, apparently including the Intel compiler. The settlement says:

> 2.3 TECHNICAL PRACTICES
>
> Intel shall not include any Artificial Performance Impairment in any Intel product or require any Third Party to include an Artificial Performance Impairment in the Third Party's product. As used in this Section 2.3, "Artificial Performance Impairment" means an affirmative engineering or design action by Intel (but not a failure to act) that (i) degrades the performance or operation of a Specified AMD product, (ii) is not a consequence of an Intel Product Benefit and (iii) is made intentionally to degrade the performance or operation of a Specified AMD Product. For purposes of this Section 2.3, "Product Benefit" shall mean any benefit, advantage, or improvement in terms of performance, operation, price, cost, manufacturability, reliability, compatibility, or ability to operate or enhance the operation of another product.
>
> In no circumstances shall this Section 2.3 impose or be construed to impose any obligation on Intel to (i) take any act that would provide a Product Benefit to any AMD or other non-Intel product, either when such AMD or non-Intel product is used alone or in combination with any other product, (ii) optimize any products for Specified AMD Products, or (iii) provide any technical information, documents, or know how to AMD.

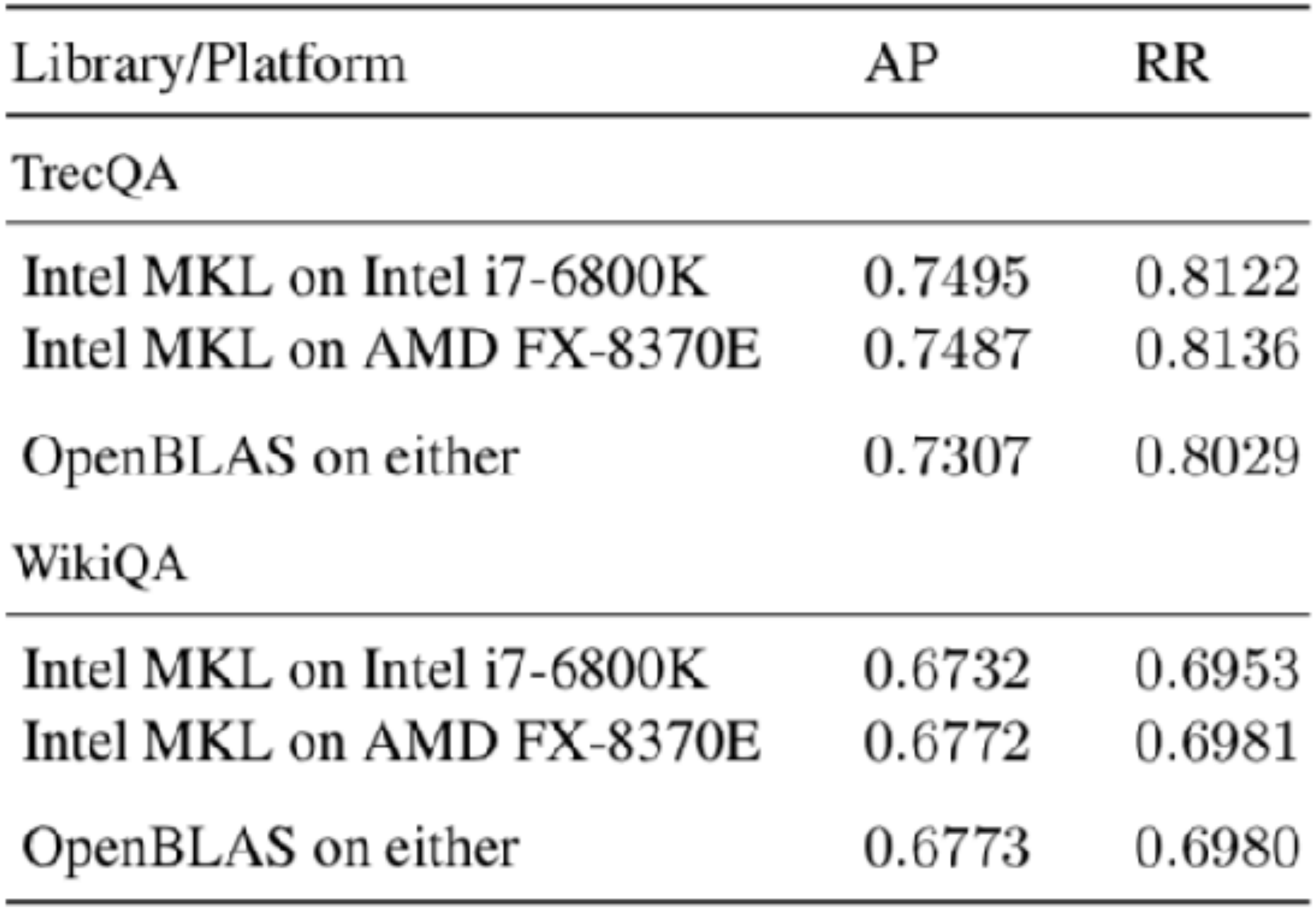
This looks like a victory for AMD. If we read "any Intel product" as Intel's compilers and function libraries, "any Third Party" as programmers using these compilers and libraries, and "Artificial Performance Impairment" as the CPU dispatcher checking the vendor ID string; then the settlement puts an obligation on Intel to change their CPU dispatcher. I will certainly check the next version of Intel's compiler and libraries to see if they have done so or they have found a loophole in the settlement.

Interestingly, this is not the end of the story. Only about one month after the AMD/Intel settlement, the US Federal Trade Commission (FTC) filed an antitrust complaint against Intel. The accusations in the FTC complaint are unusually strong:

> Intel sought to undercut the performance advantage of non-Intel x86 CPUs relative to Intel x86 CPUs when it redesigned and distributed software products, such as compilers and libraries.
> [...]
> To the public, OEMs, ISVs, and benchmarking organizations, the slower performance of non-Intel CPUs on Intel-compiled software applications appeared to be caused by the non-Intel CPUs rather than the Intel software. Intel failed to disclose the effects of the changes it made to its software in or about 2003 and later to its customers or the public. Intel also disseminated false or misleading documentation about its compiler and libraries. Intel represented to ISVs, OEMs, benchmarking organizations, and the public that programs inherently performed better on Intel CPUs than on competing CPUs. In truth and in fact, many differences were due largely or entirely to the Intel software. Intel's misleading or false statements and omissions about the performance of its software were material to ISVs, OEMs, benchmarking organizations, and the public in their purchase or use of CPUs. Therefore, Intel's representations that programs inherently performed better on Intel CPUs than on competing CPUs were, and are, false or misleading. Intel's failure to disclose that the differences were due largely to the Intel software, in light of the representations made, was, and is, a deceptive practice. Moreover, those misrepresentations and omissions were likely to harm the reputation of other x86 CPUs companies, and harmed competition.
> [...]
> Some ISVs requested information from Intel concerning the apparent variation in performance of identical software run on Intel and non-Intel CPUs. In response to such requests, on numerous occasions, Intel misrepresented, expressly or by implication, the source of the problem and whether it could be solved.

http://agner.org/optimize/blog/read.php?i=49#49

Fixing To A Neutral Math Library



Library/Platform	AP	RR
TrecQA		
Intel MKL on Intel i7-6800K	0.7495	0.8122
Intel MKL on AMD FX-8370E	0.7487	0.8136
OpenBLAS on either	0.7307	0.8029
WikiQA		
Intel MKL on Intel i7-6800K	0.6732	0.6953
Intel MKL on AMD FX-8370E	0.6772	0.6981
OpenBLAS on either	0.6773	0.6980

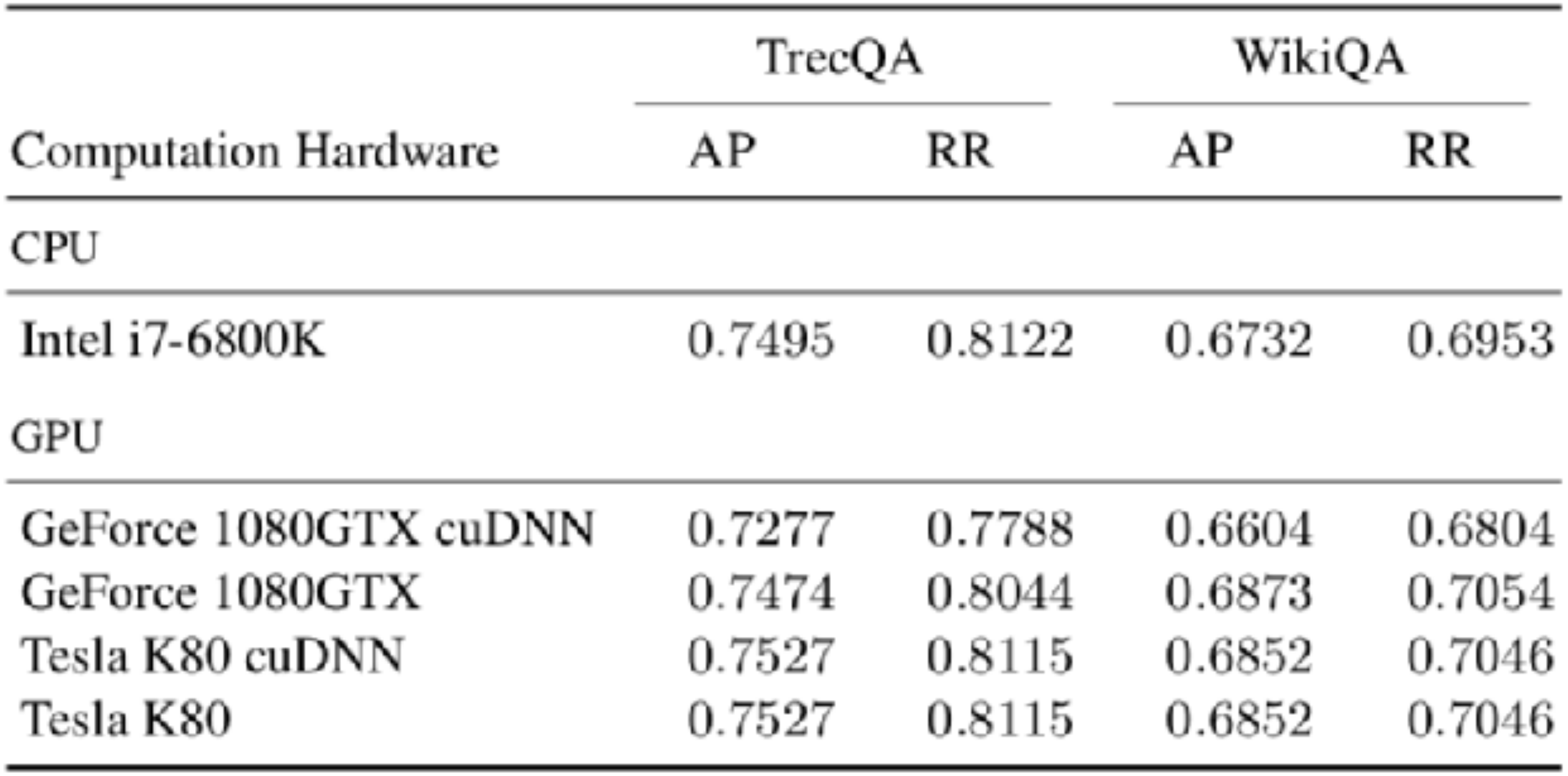
Where Are We?

- Fully reproducible, replicable training of networks on the CPU

- Fix:
  - version of model definition
  - version of framework
  - version of framework dependencies (not investigated, but… duh)
  - number of threads
  - framework dependencies to be non-hardware specific

What about GPU?

- Bajillion's of different GPUs out there, and have very little control over some aspects, as an example, we can’t fix the number of threads



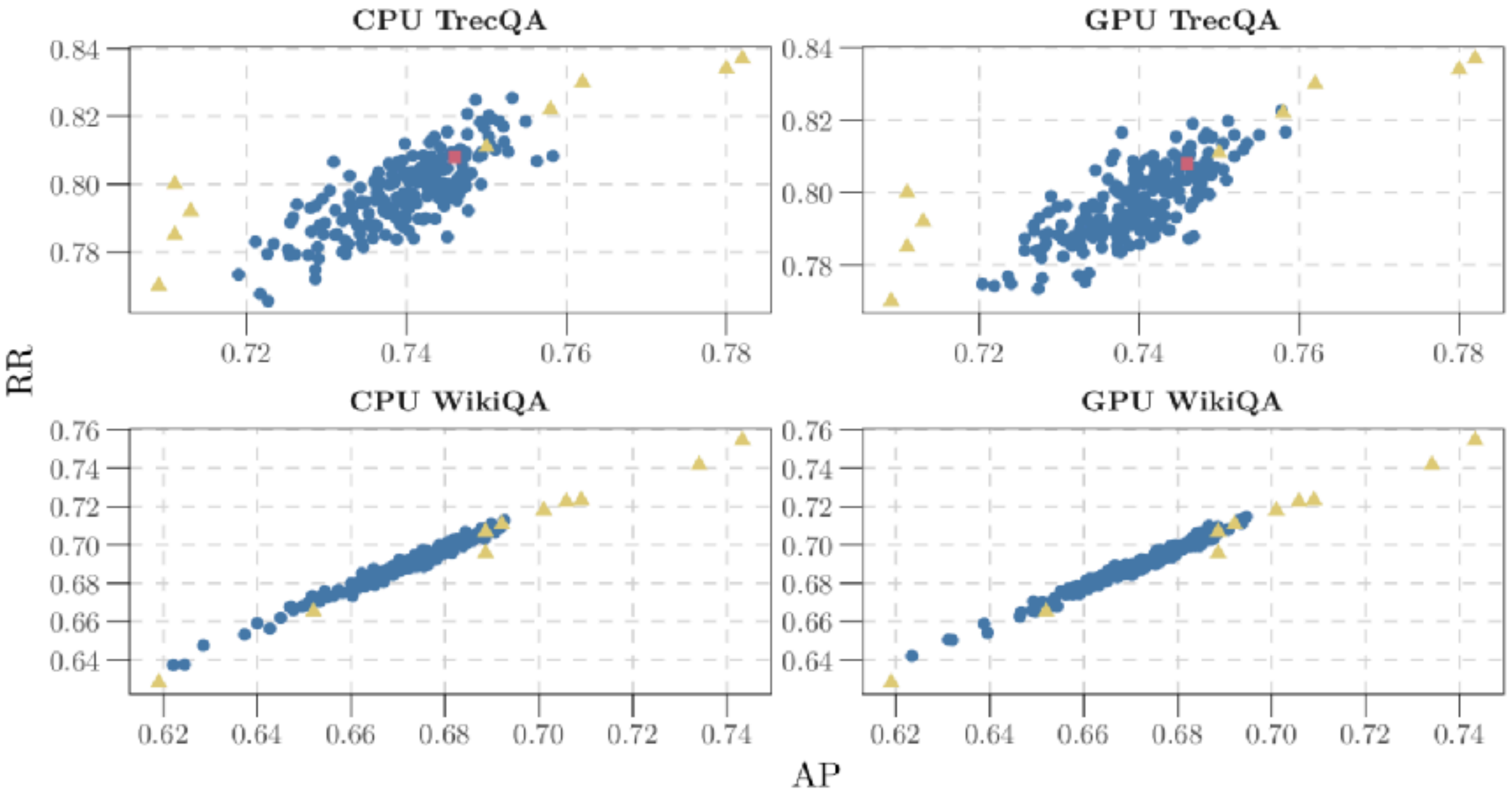
	TrecQA		WikiQA	
Computation Hardware	AP	RR	AP	RR
CPU				
Intel i7-6800K	0.7495	0.8122	0.6732	0.6953
GPU				
GeForce 1080GTX cuDNN	0.7277	0.7788	0.6604	0.6804
GeForce 1080GTX	0.7474	0.8044	0.6873	0.7054
Tesla K80 cuDNN	0.7527	0.8115	0.6852	0.7046
Tesla K80	0.7527	0.8115	0.6852	0.7046

- cuDNN? Enable or disable the cuDNN backend as shipped by nVidia. Has (potentially) non-reproducible kernels.

Nathan Whitehead and Alex Fit-Florea. 2011. Precision & Performance: Floating Point and IEEE 754 Compliance for NVIDIA GPUs.

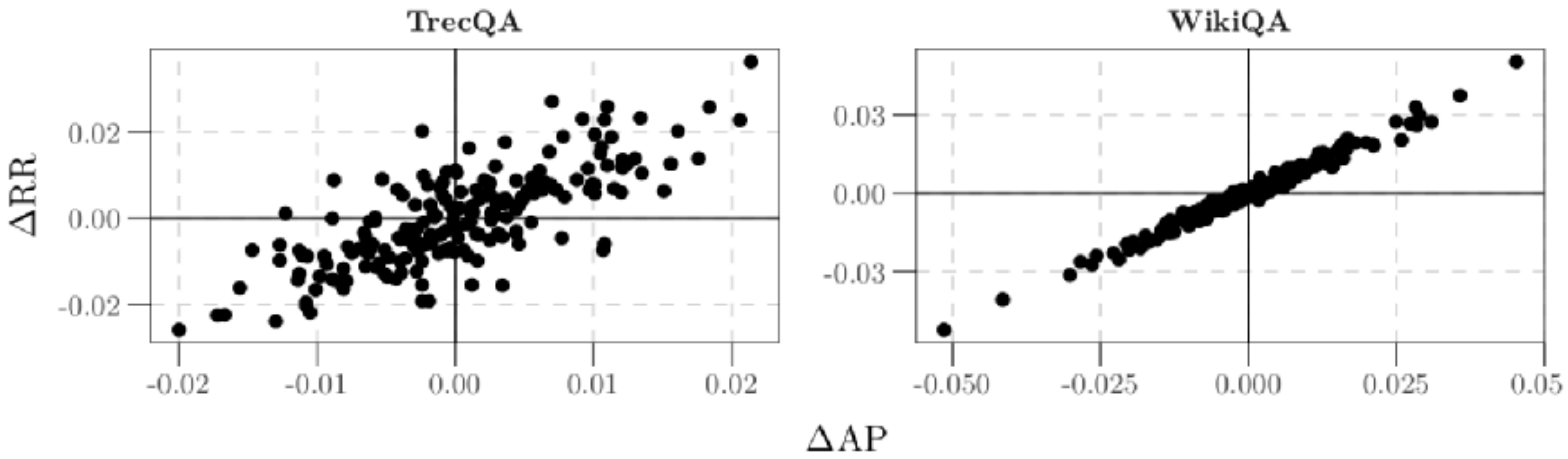
You Reap What You Sow

- What about the random seed? Maybe we got lucky, and chose a good seed?

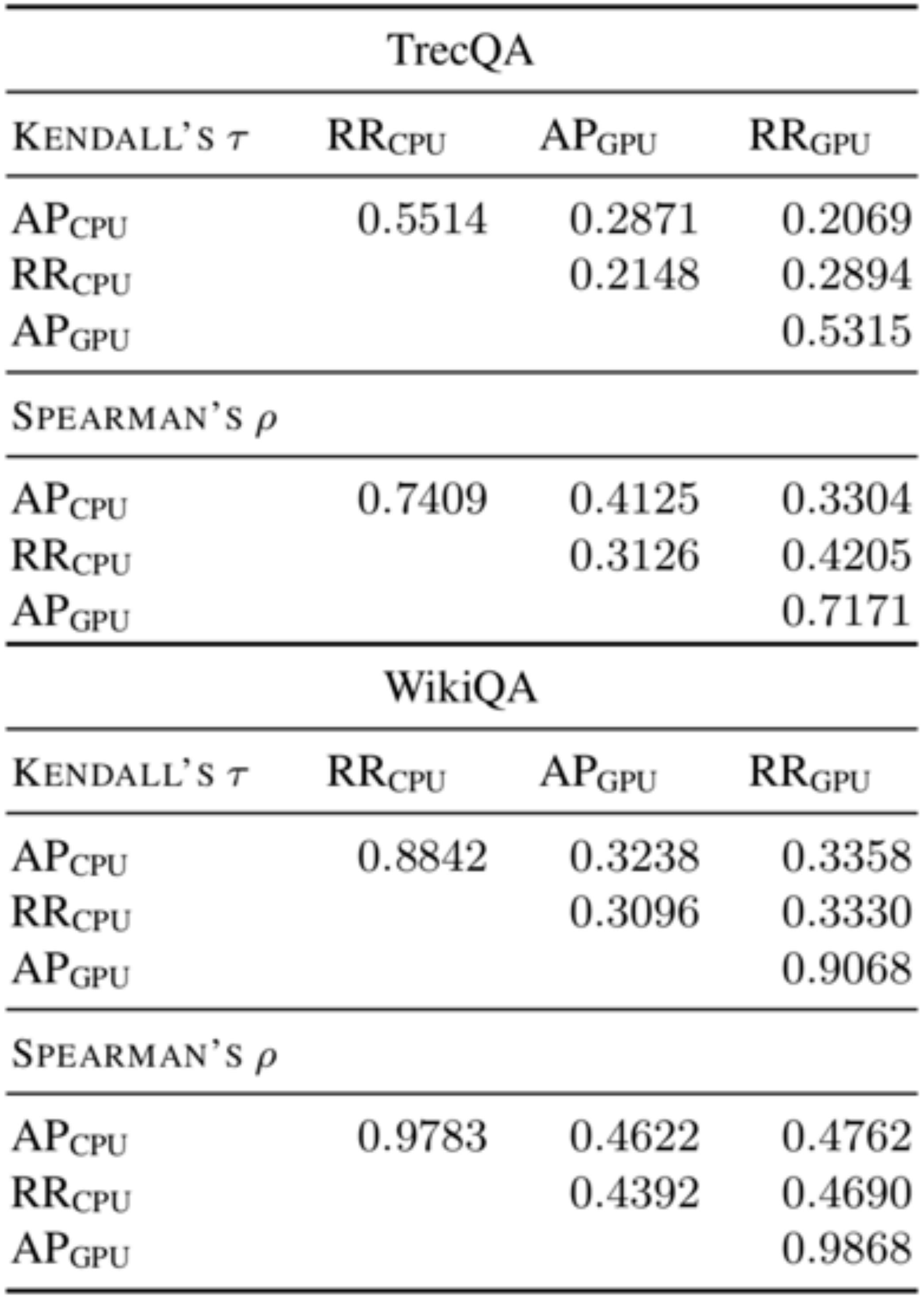


At Least CPU & GPU are Roughly Equivalent?

- but…



Agree to Disagree



TrecQA

KENDALL'S τ	RR_{CPU}	AP_{GPU}	RR_{GPU}
AP_{CPU}	0.5514	0.2871	0.2069
RR_{CPU}		0.2148	0.2894
AP_{GPU}			0.5315
SPEARMAN'S ρ			
AP_{CPU}	0.7409	0.4125	0.3304
RR_{CPU}		0.3126	0.4205
AP_{GPU}			0.7171

WikiQA

KENDALL'S τ	RR_{CPU}	AP_{GPU}	RR_{GPU}
AP_{CPU}	0.8842	0.3238	0.3358
RR_{CPU}		0.3096	0.3330
AP_{GPU}			0.9068
SPEARMAN'S ρ			
AP_{CPU}	0.9783	0.4622	0.4762
RR_{CPU}		0.4392	0.4690
AP_{GPU}			0.9868

- Only moderate-strong agreement between metrics when trained on the same hardware
- Weak agreement between metrics when trained on different hardware
- Weak agreement on the same metric when trained on different hardware

Conclusions

- All these things make a difference, nobody reports them
- Nothing to really be done, if you don’t have the same hardware, then you can’t exactly reproduce the results — but at least you can compare with that caveat
- Pre-trained models are consistent — but only marginally better than believing numbers reported in a paper
- Stop reporting single numbers, report populations

Time to Answer Questions about Questionable Answers in Question Answering Research