Twitter Thread by Kieran Snyder



Kieran Snyder
@KieranSnyder

1 There's a chasm between an NLP technology that works well in the research lab and something that works for applications that real people use. This was eye-opening when I started my career, and every time I talk to an NLP engineer at @textio, it continues to strike me even now.

2 Research conditions are theoretical and/or idealized. A huge problem for so-called NLP or AI startups with highly credentialed academic founders is that they bring limited knowledge of what it takes to build real products outside the lab.

3 A product is ultimately a thing that people pay for - not just cool technology or user experience. But I'm not even talking about knowledge gaps in go-to-market work. I'm talking purely technical gaps: how you go from science project to performant + delightful user experience.

4 Most commoditized NLP packages solve well-understood problems in standard ways that sacrifice either precision or performance. In a research lab, this is not usually a hard trade-off; in general, no one is using what you make, so performance is less important than precision.

5 In software, when you're making something for real people to use, these tradeoffs are a big deal. Especially if you're asking those people to pay for what you've made (can't get away from that pesky GTM thinking). They expect quality, which includes precision AND performance.

6 Example: Let's say you're trying to do something simple and commoditized, like implement a grammar checker. (I'll pause while someone argues with me, but I stand by it: grammar checking is a commodity offering, not a commercial one.)

7 Grammar checkers have historically been rule-based, which means that someone can sit down + write a dozen/hundred/thousand rule-based statements that capture the system you want to implement. But not all rules are created equal!

8 You can choose a small number of rules that account for the majority of grammar mistakes that people make. By keeping the rule set small, you can make sure the system works faster - it won't take huge swaths of time to calculate errors and suggestions across an entire document.

9 But by choosing a small set of grammar rules, you end up with a long tail of mistakes that profoundly erodes user confidence in your system overall. You may catch 80% of the errors with 5% of the rules, but the 20% you mischaracterize makes the user think your system is trash!

10 By contrast, implementing thousands of rules gets you awesome precision. But how long do all these rules make it take to grammar-check someone's real documents? You may get all the grammar right, but your app's performance erodes user confidence anyway.

11 All this is for a "simple," commoditized feature… not so simple, even with rules, and even for something commoditized that everyone expects to "just work." Now let’s say you’re NOT implementing a grammar checker as a be-all, end-all, but as a component of a larger system.

12 The complexity that exists in your grammar checker exists across your system, and further, all the libraries you use (build, buy, or borrow) have to interact with each other… further slowing your system down and/or compromising the precision of one part in service of another.

13 You only encounter these issues as a production NLP engineer. They don’t come up in the research lab. Which is why it takes so long for great research to impact real products (which again, are things people pay for). And why so many researchers do not enjoy industry work.

14 Thanks to @kwhumphreys who inspired me thinking down this path today and who solves these problems for us every day! ■