RECOMMENDER SYSTEMS: BEST PRACTICES FOR BUILDING, DEPLOYING, AND OPTIMIZING

EXECUTIVE SUMMARY

This report builds on a series of interviews with the technical leaders of well-known businesses in retail, media, e-commerce, and more that leverage recommender systems (recsys). Participants in this study range from Tencent to The New York Times.

The intended audience for this report includes data scientists and machine learning engineers who are currently building or thinking about building recommender systems for production use cases. Core objectives include contributing to the wider industry conversation by providing helpful insights from experts in the field and articulating best practices for building, deploying, and optimizing recommender systems.

> Core assumptions which have been confirmed through this study include:

1. The process of building relevant recommenders is hard
2. Industry-wide openness to sharing what works best is crucial for the field to advance

> Consequently, the report has been structured to show:

1. A preview of the trends for recommender systems practices in industry
2. A brief history of recommender systems, illustrating the evolution from 1979 to 2009, from academic experiments to large-scale commercial successes
3. Selected excerpts from the expert interviews
4. A summary of the observed trends, with some indication of future trajectories
5. A selection of in-depth interviews, which explore the context and nuances of recsys practices

A PREVIEW OF RECOMMENDER SYSTEMS TRENDS

It almost goes without saying that open source is not merely an option for recommender systems in industry. It's become table stakes. The interoperability of tools within an open source ecosystem is crucial for de-risking projects, especially when considered over their full lifecycle. This tooling must be flexible enough to support experimentation and exploration, while admitting new technologies that haven't yet emerged.

The other given here is the matter of data. Since notions of "big data" emerged in the mid-2000s, one truism about machine learning is that having more data—and better-quality data—tends to result in better models. Of course, this holds for recommender systems as well. We cannot depend solely on having superior algorithms to produce models; we must have good training data, good customer data for inference, good feedback and instrumentation for evaluating metrics, and so on. Priorities for effective data preparation will likely be a perpetual requirement. Also, data rates will grow as a recommender use case gains success. There's also the matter of learning from the data as metrics are refined for success. It may even help to reframe your data science staff as "metrics engineers" for the sake of thoroughly understanding recommenders in production.

> **Beyond these two givens, a few other important points emerge from this study:**

1. For those starting their recsys journey, the experts were nearly unanimous in their advice: In lieu of rushing to implement complex machine learning models which may be trendy, start simple and ideate about what's needed.
2. It's essential to understand the metrics used in production and thoroughly understand the objective function for the use case at hand.
3. Although recommenders used to be thought of as point solutions, they're extending deeper into business operations. Verticals with historically thin margins are now using recommenders to grow their businesses while increasing customer trust and loyalty.
4. Many production use cases for recommender systems have fast inference requirements, typically less than 100 milliseconds. As more advanced AI technologies become available, will they run fast enough to fit within those constraints? Hardware acceleration becomes the enabling factor.
5. The hot topics related to recommender practices in industry include using feature stores, making use of graph neural networks, and leveraging model distillation.
6. Lessons learned: Plan for a roadmap of system upgrades and growing use cases. Where possible, leverage technologies that provide a path forward, not merely for the scaling requirements today but for what your successful business will need a few years ahead

RECOMMENDER SYSTEMS: ORIGIN STORIES

Looking at recommender systems practices in industry, there's a lot of common ground—shared challenges, shared concerns, common areas of forward-looking research. Similarly, looking at the history of recommender systems, we can identify themes that would repeat over the decades, as new teams began their respective journeys and began learning from them.

This section provides a retrospective on notable projects that helped establish the field of recommenders. In particular, it traces the evolution from early academic experiments through large-scale commercial successes. We can learn from the patterns of the past and also use this history as a lens for comparing issues and themes among contemporary practices.

Before the 1990s: Grundy, Deep Models, and Books

The notion of recommender systems running online began to emerge in the 1990s, tracking closely alongside the growth of the World Wide Web in general. We can look back to an early project called Grundy by Elaine Rich, which originated in her PhD work at Carnegie Mellon University in 1979 and continued in her faculty role at the University of Texas at Austin. Rich explored the use of stereotypes to build and generalize user models. The system would suggest novels to a user (as a librarian might) and then the user would provide feedback about the quality of the recommendations. This work was during relatively early days for machine learning, although some of the key elements were in place, including feature engineering, data training, model evaluation, and feedback for iterative improvements to the model. Considering the prevailing views in the AI community during the late 1970s, the Grundy system relied on relatively deep information about each user, while attempting to minimize user engagement with the system.

1990s: The Rise of Collaborative Filtering

Doug Terry is credited with creating the first collaborative filtering recommender system, called Tapestry, at Xerox PARC in 1992. This project addressed the emerging ills at the time of email overload. Users would annotate email messages to note their interests, and the system would learn from these annotations. Terry explained in the original paper about Tapestry that its primary technical innovation was an efficient algorithm for filter queries which had predictable semantics. Already by this point in 1992, Terry identified key points going forward about collaborative filtering: the need for lots of training data, concerns about security and privacy, and the fact that Tapestry had not yet been integrated into any of the new "web browser" software.

Two years later, Paul Resnik and others established the GroupLens project at the Massachusetts Institute of Technology (MIT) in 1994, which continues now at the University of Minnesota. Their work embellished on Terry's Tapestry project to provide collaborative filtering for online newsgroups content, recognizing that, while people read and reacted to news articles online, the data was largely ignored. In contrast to Elaine Rich's perspectives about Grundy, the work on GroupLens leveraged relatively shallow data and lots of it. They also used an open protocol, so that anyone could modify a news client to integrate ratings and predicted scores.

MIT, Firefly Music Recommendations, and E-Commerce

Interest in recommender systems spread at MIT, with professor Pattie Maes and graduate student Upendra Shardanand developing the Ringo system for music recommendations. Notably, this system featured more personalized recommendations and scaled to over 2,000 users. The Ringo authors referenced Rich's stereotypes for user modeling and used benchmarks to show that their core algorithm outperformed GroupLens by Resnik and company. They also noted the need for use of machine learning going forward, for example, using clustering algorithms on the data to help speed up the recommendation results and experiment with content modeling.

In a following project, Pattie Maes and others launched the Firefly music recommender system at MIT in 1995. This spun out as a venture-backed startup and was subsequently acquired by Microsoft in 1998. The company expanded beyond music recommendations into other content such as books, news, etc., and licensed their technology out to several influential firms during this early era of the internet, including Yahoo!, ZDNet, Barnes & Noble, America Online, and Reuters.

Arguably, Firefly represented the first recommender system engaged in e-commerce that leveraged machine learning. It would be the first among the early recsys research projects to gain commercial success and is the most recognizable by today's notions of social media and e-commerce content recommendation.

Netflix and Amazon Content Recommendations

Meanwhile, as a new startup in 1998, Netflix challenged the in-person video rental market led by Blockbuster, offering a DVD-by-mail subscription. In 2000, Netflix shifted their business model to an "all you can eat" subscription service instead, extending a much larger content inventory, and added member ratings for movies and a personalized recommendations service called Cinematch based on collaborative filtering. In 2001, the company introduced a five-star rating to drive recommendations further, followed by an ecosystem of other algorithms to help personalize and merchandise the process of movie rentals.

Amazon was also experimenting with content recommendations during this time. The firm had famously "split the website" in mid-1997 in an effort led by Greg Linden, which pioneered horizontal scale-out and practices that would later become cloud computing. This led to a dramatic ramp in the collection of machine data at scale to characterize customer engagement, which Andrew Ng and others have called the "virtuous cycle" of data and machine learning. In 2003, Amazon published the famous paper "Amazon.com Recommendations: Item-to-Item Collaborative Filtering" by Greg Linden, Brent Smith, and Jerry York. By this point in 2003, the authors noted that there had been three common approaches for content discovery and recommendations: collaborative filtering, clustering models (for example, for dimensional reduction), and search. Their work differed by introducing item-to-item collaborative filtering, which led to the well-known phrase "People who bought this book also bought..." In other words, Amazon's recommender system could scale independently of the number of users or the number of content items. They prioritized fast inference to produce recommendations on the wire, massive datasets, and quality measures for evaluating the success of recommendations.

Amazon of course set the pace for recommender-driven e-commerce as it continued to scale and build more business lines. Direct outcomes from this work also included the origins of cloud computing, big data practices, and commercial use cases for machine learning.

Recommender Systems and Social Networks

A number of social networks followed, applying similar approaches to what became a new generation of communication platforms. Among these, LinkedIn was scaling up in 2006 when Jonathan Goldman joined the firm. Recognizing the value of network analytics, Goldman built a recommender system for "People You May Know" (PYMK), despite disinterest by product teams. This proved to be quite popular among LinkedIn's users. Click-through rates (CTRs) for PYMK-based content initially ran 30 percent higher than other approaches, and recommendations rapidly became an essential component of user experience at this business-oriented social network.

By this point in the late 2000s, the popularity of recommender systems had become relatively commonplace in production use. Other kinds of businesses were making great use of the technology, such as online dating systems, online gaming, and retail. The term data science emerged into vernacular around 2009, with mainstream firms adopting the practice and top talent getting attracted to the field. Arguably, recommender systems and their value in e-commerce played no small role in that evolution.

The Netflix Prize

From 2006 through 2009, Netflix curated a competition for recommender systems called the Netflix Prize where teams competed on a leaderboard to optimize recommendations using deidentified customer ratings of movies. While competition entries significantly outperformed the Netflix algorithms, for a variety of reasons, these entries were never put into production. Even so, the experiences of the leading teams in the competition revealed a wealth of learnings about comparative strategies for building recommender systems, analyzing the pitfalls of ratings data, and ultimately the value of ensembles for machine learning work in production.

THE EVOLUTION OF RECOMMENDER SYSTEMS

Within less than two decades, recommender systems had evolved from being a toy project that helped manage a researcher's email inbox to being core technologies that drive the revenues of leading technology firms. Key concepts and requirements for recommender systems identified through these early projects include:

- Collaborative filtering
- Personalization
- Use of massive datasets, without especially good data
- Leveraging distributed systems for scale
- Scaling independently of number of users or items
- User modeling, content modeling, and other machine learning within the same use case
- Quantitative evaluation of recommendation quality
- Security and data privacy concerns
- The need for fast inference
- How difficult good recommendations are in practice
- Tying recommender quality to business revenue
- Metrics for success, e.g., LinkedIn's CTR lift from the PYMK introduction

OBSERVATIONS

This history brings us up to the beginning of the previous decade. A few key observations about opportunities, challenges, and navigation emerge from the history of recommender system, which we'll use later to review current trends.

> Opportunities: User Engagement

The first observation is about opportunities. Wherever a community of users is engaged in an online activity, both the implicit data exhaust and the explicit feedback from those users can probably be generalized to enhance the user experience. Recommender systems have been used in many ways to enhance user experience across a wide range of activities, including e-commerce, online dating, news readers, sales leads, gaming, music apps, and so on. The broader scope of this problem is almost always about discovery, and an e-commerce scenario can illustrate the most common modes for it:

1. **Navigation:** When a customer knows exactly which widget they need to buy, help them navigate to that specific widget's page, and then check out. In practice, this kind of "pull" condition is relatively rare, although important for optimizing UX design.
2. **Search:** When a customer knows approximately which widget they need to buy, help them search for items that are close, and let them select among a collection of results. Of course, search is widespread and historically simpler to implement and introduce into UX in general.
3. **Recommenders:** When a customer is browsing or engaged in another activity, "push" recommended items to them by personalizing their web browsing experience. This customer scenario is statistically more common, although more complex and nuanced to implement effectively than the previous two UX scenarios.

For example, looking at the history of how the Netflix business model evolved, their UX progressed through these stages. Initially in 1998, their UX relied on navigation plus some search. Quickly that followed with recommendations and personalization being introduced. Over the years, more and more levels of personalization have been added, powered by more distinct use cases for recommender systems. Adoption took time though: Initially 20 percent of selected content at Netflix had been recommended, which ramped up to 80 percent over time.

This kind of arc shows up where organizations have been adopting AI applications to enhance their online business activities. Start simple, and add increasingly more sophisticated and nuanced approaches in stages. Ultimately there won't be just one recommender employed for a given use case, but probably several. Throughout this progression, the volume of engagement that depends on recommendations shifts from "a little" to "most" of the user experience.

> Challenges: Recommendation Quality

The second observation is about challenges: It's very difficult to get a proper recommender system working effectively. Good data to use for training models is hard to obtain. People change their preferences dramatically. For example, one of the highest-valued categories of ad placements is for high-end automobiles. Recommendations for new luxury car sales tend to perform well in a bid/ask advertising network. In other words, a person who would like to purchase a new luxury car tends to click on related ads, and that translates to value. However, once an individual has purchased their new car, it's likely that the last thing they'll do is click on related ads. A recommendation that had been consistently good for a period of time suddenly becomes bad. Lacking information about who has purchased what and when, recommender systems for advertising have inherent risks about quality—and this problem extends into most application areas for recommender systems.

Complaints about the quality of recommendations date back to Elaine Rich's Grundy system in 1979 and have followed ever since. While it's feasible to recommend some items to a given user at a given point, it's incredibly challenging to recommend precisely the items that a given user would want at a given point—while avoiding any items that they don't want. Even so, these are the criteria whereby users tend to judge automated systems in general.

Moreover, in a kind of double-edged sword, when recommendations become too good some consider the user experience creepy. The notion of the uncanny valley describes a problem in aesthetics and design where automation gets perceived as encroaching on humanness, which in turn causes revulsion among observers. This can pose a dilemma in that good recommendations get perceived as the system knowing "too much" about users. Considerations about data privacy in recommender systems date back to Doug Terry at Xerox PARC in 1992.

Of course, machine learning has played such an important role throughout the history of recommender systems, and machine learning as a field was undergoing rapid evolution during this same period. Consider that machine learning is largely about using historic or synthesized data to generalize patterns which can then be applied at a later time or in different contexts. That's an almost ideal fit with the needs of recommender systems in practice. One catch is that, as machine learning models generalize from training data, they also tend to lose context. It may be difficult to explain why a particular set of recommendations have been made for a given user. This can pose extreme challenges when data teams need to work on recommender quality or troubleshoot workflows that use machine learning models.

> Narratives: Best Practices for Recommender System Use Cases

A third observation is about narratives. Throughout this early growth period, the general perception of recommender technologies at any point was driven by narratives from a handful of companies. While interesting from a technology perspective and often regarded as "best practices" by industry analysts, these narratives about large use cases were not always the best advice for recommender system use cases in general.

Early research work tended to view recommenders in terms of information retrieval, where a model was only as good as its precision and recall metrics. One consequence was that Elaine Rich's depth of user modeling was all but forgotten. Also, this academic perspective was not especially effective for troubleshooting and tuning complex systems in production at scale.

Commercial successes at Netflix and similar practices later emphasized the use of non-negative matrix factorization, a narrative that fit well with the contemporary popularity of big data tools such as Apache Hadoop. More recently, as deep learning has become popular, recommender system practices have turned toward neural networks and embeddings.

These narratives and trends serve a purpose in terms of helping spread innovations from leading technology firms out across a broader audience. They also help trace the evolution of thinking about data science in general, as demand grew for addressing increasingly complex challenges with more sophisticated approaches. However, these kinds of narratives also tend to confound the underlying drivers and tradeoffs that make for good recommender system practices. We'll explore what these drivers and tradeoffs are in more detail in the following sections.

Overall, for the results of recommendation systems to be effective, clearly a data team must balance several competing priorities and concerns. This is no simple task. Tensions exist among these kinds of tradeoffs in recommender systems practices, some of which will probably continue to be concerns for the foreseeable future. Some issues can be addressed through better tooling. Next we'll consider how recommender systems teams in industry approach their practices.

INTERVIEWS: KEY EXCERPTS

We've worked with select teams and individuals in industry who are experts in recommender systems, talking with the technical leads responsible for recommender workflows. For the sake of brevity, the following sections excerpt key points from the full interviews.

During each interview, we explored questions about the history of their team, their team focus and responsibilities, and how they handle training for their team. We asked specifically about their recommender systems practice in terms of data rates and scale, approaches used for data preparation and feature engineering, how they select appropriate technologies and frameworks, means for evaluating models and tuning them, and what they do to optimize recommenders. We also asked about general advice for those just starting out on a recommender systems journey.

Let's explore the common ground among these interviews and areas of contrast among different practices in industry. Where possible, we'll use key points from recsys history as a lens for this analysis.

Monica Rogati, Independent Advisor

AI and Data Science Advisor and creator of the first machine learning model for LinkedIn's PYMK recommender

Monica Rogati has a PhD in computer science from CMU, and her background includes applied machine learning , natural language processing (NLP), and wearables. She was an early member of the LinkedIn data science team working on recommenders and other data products, then became vice president of data at Jawbone where she built and led an expert team of data scientists and engineers. Currently an independent advisor, Monica is widely cited for the "AI Pyramid of Needs" diagram, which serves as a shield for misunderstood data scientists everywhere.

In terms of scale and data rates for recommenders, Monica has built systems that varied from zero data to matching hundreds of million users to as many items. As Monica explains, "The upper end of this range gets most of the attention as it can address the long tail and make very personalized recommendations that feel like magic (or creepy, depending on your perspective)."

> Key Challenges in Recommender Systems

What are the major challenges in practice for recommender systems? One of the biggest challenges is to determine the appropriate metrics and objective function for a given use case. **"Data scientists will eventually become 'metric engineers.' It's one of the data tasks that is hardest, if not impossible, to automate."**

Monica cautions practitioners to think carefully about the metrics they're trying to maximize, i.e., the recommender's objective function. This represents a proxy for what the use case is attempting to achieve. "Will it still be a good proxy after you influenced it with your system? Is there a way to game it, willingly or inadvertently due to incentive alignment? How hard is the problem and what's the upper limit of what you'd expect as far as performance goes? For example, what's the inter-annotator agreement on your training data?"

Monica also advises about selecting appropriate technologies for building recommenders. "There are a few criteria I keep in mind, in addition to the likelihood of solving the problem: ease of integration, maturity, and preserving optionality. One of the most important qualities is ease of integration with a company's current workflow and keeping the ratio of effort to results as small as possible."

"In the past few years, there's been an explosion of data tools and frameworks, and I'm happy to see that many of them do have easy integration points that don't require significant changes to the workflow until the tool is proven and trusted in a particular environment. My go-to question for people building these tools is 'What's your Trojan Horse? What is the one problem you could solve with low or no effort on the practitioner's part to gain adoption and trust?'"

The point is to de-risk technology selections early in the development lifecycle and have backup plans for contingencies. "The best tools include possible exit paths in their marketing and provide data export tools in addition to easy integration points." This tends to fit well with open source: When "ease of integration" is prioritized, replacing one component allows for "preserving options" while minimizing disruption and keeping that ratio of effort to results.

> Initiating Recommender Projects

"Something I learned from Salim Roukos while interning at IBM Research: Before starting to work on a machine learning system, spend an hour labeling data and doing the algorithm's job. Be the algorithm. You'll find countless data quality issues, identify a set of situations when the question is ill-defined, or when you should abstain from making a recommendation."

To evaluate models in general, Monica Rogati describes a three-step process which also helps de-risk projects:

1. Create a simple baseline
2. Build a lenient offline filter
3. A/B testing with slow ramp up

This first step is to analyze the data then implement a reasonable but simple rule-based baseline. Even if you already have a model in place, try this anyway. Simple baselines are easy to debug, robust, fast, and they should be preferred when they're almost as good as your current model.

Using a "lenient offline filter" provides a twist on traditional notions of offline model evaluation. Check whether a model would result in huge disruptions or unexpected harm and whether its results appear promising in a loose sense of the word. The goal isn't necessarily to outperform an existing system—although if you started above with a simple baseline, it should.

Finally, run a true evaluation of a trained machine learning model using A/B testing in production but with a very slow ramp up. For example, 1 percent, then 2 percent, 5 percent, 10 percent, 20 percent, and so on, confirming its metrics at each step. Watch for operational issues, missing features, unexpected harm, and generally any problems that might not appear offline or at a small scale.

After a machine learning model is running at full scale, especially for recommender systems, often there will need to be a "burn in" period to allow any novelty effects to wear off.

Xiangting Kong, Tencent

Expert engineer at Tencent, responsible for the design and development of the advertising recommendation system, and the lead for the Tencent Advertising and Deep Learning Platform

Xiangting Kong leads the team at Tencent focused on the advertising recommender system, which is responsible for optimization of the advertising training platform. Components of this platform include offline feature engineering, training platform, online inference system, online feature engineering, and the play platform. They work on model training, optimization, and inference in a variety of business scenarios, ranging from advertising and fintech to networking data mining. Given their range of responsibilities and the technologies involved, for team training they organize technology sharing sessions every one to two weeks.

Xiangting advises, "Advertising recommendations is a process of gradual filtering. Sorting stages include recall, pre-ranking, and ranking. Each stage has different requirements. The rapid investigation and iteration of the model puts forth higher requirements for training performance."

There are important tradeoffs to consider regarding data rates and model complexity. On the one hand, increased volume of training/testing data and more complex feature sets can improve model quality. On the other hand, these also tend to increase model training times and limit model refresh rates. **"The accuracy of the advertisement recommendation can be improved by training more sample data, by adding more sample features. But this leads to longer training time and affects the update frequency of the model."**

The team at Tencent uses HugeCTR for their recommendation training framework. This has been integrated into their advertising recommender platform to accelerate the frequency of model training and updates. Also, more data samples can then be used during training to help improve the results of recommendations. In terms of technology selections in general, Xiangting describes priorities for working with open source projects. "The technology or framework we choose must be compatible with the community ecosystem, so that we can do better follow-up upgrades."

> Leveraging Recent Research

In terms of current research being leveraged in their practice, Tencent recently integrated a compressed sparse row (CSR) pipeline into their ad recommender training. "CSR-type training data is generated so that the data can be directly read on the GPU for training. Through our optimization of the data processing pipeline, the CPU load is greatly reduced and the GPU utilization is greatly improved."

As general advice for teams just starting their recommender system journey, Xiangting advises a steady course in the face of a rapidly evolving landscape. "Choose a mature technical framework and be compatible with the community ecosystem to facilitate subsequent system upgrades."

Felipe Contratres, Magalu

Tribe Leader for the Personalization Team at Magalu (Magazine Luiza)

Felipe Contratres leads Magalu's personalization team, which runs the recommendation and search platforms, creating new functionalities for their platform and new recommendations models. For those who are just starting out with recommenders, Felipe advises, "Start with simple approaches: Many problems are easily handled by simple techniques that reduce your overall system complexity. When more advanced techniques are necessary, choose mature packages or frameworks to integrate into your workflow." Felipe also recommends testing a new solution in a production environment as soon as possible. "Try to put your model online as fast as possible to test it with real customers instead of trying to pre-optimize excessively offline."

Their team has been looking at multi-modal features for combining tabular events with textual and image vectors. "We are considering testing the technique proposed by Gabriel de Souza P. Moreira, et al., in the recent paper." They also have a heuristic session-based recommender that customizes the user experience between sessions. This current session-based recommender was created based on prior research combined with Magalu's business knowledge.

Jun Huang, Meituan

Senior Technical Expert at Meituan, responsible for the training framework team of the Meituan Machine Learning Platform.

According to Meituan, "Our mission is: 'We Help People Eat Better, Live Better.'" As China's leading e-commerce platform for services, Meituan's business revolves around the "Food+ Platform" strategy, and is centered on "eating" as its core. Meituan operates several well-known mobile apps in China, including Meituan, Dianping, Meituan Waimai and others. Its business comprises over 200 service categories, including catering, on-demand delivery, car-hailing, bike-sharing, hotel and travel booking, movie ticketing, and other entertainment and lifestyle services, covering over 2,800 cities and counties across China.

The training framework team at Meituan produces a high-performance distributed deep learning training framework deployed in large-scale CPU/GPU clusters. For recommenders, this supports distributed training with ~100 billion sparse parameters and ~100 billion samples. Jun explains, "Recently, we have designed the next generation of recommender training system based on the NVIDIA A100, which greatly improves the training efficiency and the model complexity."

In terms of choosing appropriate technologies, Jun describes the priorities placed on open source. "This technology needs to be advanced, open, and ecological, so that we can fulfill our internal needs better based on it. Our team is currently building our system mainly based on open source technology. At the same time, we are very happy to give back our work to the open source community."

Hardware plays a vital role in their strategy. Jun explains, "At first, we optimized the training framework based on CPU architecture, but as our models became more and more complex, it was difficult to optimize the training framework deeply. Now, we are working on integrating NVIDIA HugeCTR into our training system based on A100 GPUs. A single server with 8x A100 GPUs can replace hundreds of workers in the CPU-based training system."

For those just starting with recommenders, Jun advises, "Fully understand the current company's infrastructure and business status, and design systems and processes based on it. When choosing the technology stack and framework, consider the maturity, community ecology, extensibility, and integration friendliness of each system."

Chris Wiggins, The New York Times

Chief Data Scientist, The New York Times.

Chris Wiggins leads the data science team at The New York Times, which is responsible for developing and deploying machine learning solutions for newsroom and business problems.

In terms of the role that recommender systems play in contemporary news publishing, Chris explains, "Subscribers are interested in The Times' editorial judgment, so **recommendation means optimizing for trust and loyalty and ensuring that we do so in a way that scales editorial judgment rather than replacing it."** Recommenders help augment a wide variety of different news services, including "Editor Picks," "Most Popular," "Smarter Living," the Cooking app, the "For You" personalization tab in the mobile app, the "Your Weekly Edition" newsletter, and more.

Chris advises that, in terms of choosing appropriate technology, he uses Kreps Law—named after Apache Kafka co-author Jay Kreps. "Trick for productionizing research: read current 3-5 pubs and note the stupid simple thing they all claim to beat, implement that."

However in contrast, "As a tactic, it depends a lot on the goals of our product and newsroom partners, who are very thoughtful about the 'x' (the context), the 'a' (what possible actions or articles or assets are under consideration to recommend), as well as the 'y' (what outcome we're trying to optimize)."

Recommendations at The New York Times are served through a platform called Samizdat, which handles both scaling and privacy/compliance. In essence, the platform consumes information about a reader, analyzes with respect to interpreted privacy regulations, then outputs instructions for how to handle that reader's interactions. This enables their infrastructure team to adjust interpretations of data regulations across multiple regions and simplifies changes across their entire suite of products.

Their practice manages many recommender models.

For the newer teams just starting on their recommender system journey, Chris advises, "I tend to lean on the shoulders of other people's advice, e.g., Kreps Law, and of course Monica Rogati's "AI Pyramid of Needs" infographic. Some NYT-specific success has come from co-creating by close pairing between data scientists and software engineers, who sit (or sat, when we were in office) side by side, coding in a mixture of Python (especially the data scientist) and Go (especially our software engineering partners) to drive methods that are performant both in "statistics and in SLAs."

> Key Challenges in Recommender Systems

Priorities at NYT—such as building trust with their audience, models that have more depth and segmentation on the content side, etc.—inform the kind of machine learning modeling that tends to work well. "One success was moving from predictive methods to prescriptive methods, e.g., different contextual bandits which can be deployed for the separate surfaces and pools of content we power."

Personalization through recommenders complements editorial judgment, and that determines which stories lead in the news reporting. "Algorithmic curation at The Times is used in designated parts of our website and apps. We use it to select content where manual curation would be inefficient or difficult, like in the 'Smarter Living' section of the homepage or in 'Your Weekly Edition,' which is a personalized newsletter."

As Chris explains, "Using contextual bandits for article recommendations, these algorithms are great at quickly adapting to changing preferences and efficiently exploring new options. Of course in practice a sophisticated recommender system in production will have many such possible steps, so contextual bandits can be thought of as building blocks in the toolkit for constructing recommender workflows, especially for the content models that occur upstream from personalizations. "Building a simple contextual bandit can get you the performance that you need. In the news business where content churn…you want to balance predictiveness with interpretability to gain understanding."

Looking ahead, Chris notes, "We're always seeing new hills and trying to address the news landscape as it changes." He recommends staying tuned to https://open.nytimes.com as their team continues to chronicle their recommender system journey.

Kannan Achan, Walmart Global Tech

Leads the personalization and recommendations team at Walmart Global Tech

Kannan Achan leads the team responsible for personalizing the customer journey at Walmart Global Tech. "We do what is called whole-page personalization for a given page. It could be a creative banner, it could be an item carousel, it could be a CAD. Everything comes to the personalization team in one shot, and we personalize the whole page in a single shot."

Their teams are organized into three groups: understanding customers, understanding content, and online inference. This latter part involves a cross-product of customer features and content features performed at runtime. This separation of concerns gives the organization flexibility, as Kannan explains, "That really helps us to scale, collaborate, and solve a lot of interesting problems."

Recommenders use omnichannel data across the Walmart website and Walmart stores. Looking at the history of recommender use cases, the omnichannel data was initially used for analytics. Later the company recognized more needs for a richer understanding of customers, balanced with ample considerations about data privacy. "We do see a lot of customers go shop in all these properties." Part of what complicates the omnichannel data scenario—which is faced in many retail contexts that span both online and in-store—is that within the customer data there is an aspect called intent. For example, a customer may be doing their weekly shopping for groceries, while at other times, the same customer may be considering a one-off purchase such as a television. How recommender systems come into play may be quite different depending on intent. "If you think of the space of recommenders, a traditional view of recommenders is a matrix with users and ratings—i.e., collaborative filters. If you look at discovery in general, we have to provide something that is more traditional such as "People who viewed this also viewed…'; however, the moment an item goes to cart, we also have to recommend complementary items."

> Initiating Recommender Projects

In harmony with advice from Monica Rogati, Kannan advises practitioners to invest the time to ideate and really understand the objective function for a given use case. Complexities such as seasonality make understanding the context of the data even more important. "For example, people who buy tomatoes in Texas also buy jalapenos, but the same thing is not true somewhere else. They might buy a mason jar to pickle them, right?" Similarly, "Recommenders are not one-size-fits-all. Every recommender has its own objective function. It's somewhat driven by the business strategy."

From a conceptual perspective, Kannan advises a holistic approach. "Just personalizing a page alone is not enough. We just think of it as a tensor in a way, and we have to personalize the whole session as well." In this case, understanding the customer becomes priority number one. The recommender team works to understand why a customer is on the site, and then allow the customer to discover what options they have there. "That's where in the discovery funnel the recommendations really come into play, to help with conversion. More importantly, when you look at this journey, we also work on models which bring customers back to reengage."

Moreover, consider carefully what your customers are doing at each stage of their engagement. "Zero-query problems are something very fundamental in e-commerce, the only way you can handle it is with recommenders. What I mean by zero queries on the home page, where the only thing you have is customer context, your understanding of the customer, maybe a referral URL is the best you have, but somehow you have to paint a whole personalized page." In contrast, when we look at the Netflix Prize, they posed that recsys challenge as a point solution.

In terms of choosing appropriate technology, Kannan offers this advice: **"Don't approximate the problem. Approximate the solution.**" He also explains about the tradeoff between model complexity and required response times. "There is significant bearing on what can run online, because of a 40–50 millisecond budget. That really shapes the kind of technology we select. Sometimes, we are even very comfortable using a simple logistic regression or gradient boosting decision trees, because they really worked, they were predictably explainable, and we could control the latencies in a very meaningful way."

In terms of machine learning infrastructure, their team relies on open source. "We completely depend on open source. For example, PySpark, TensorFlow, Airflow, and Kubernetes are leveraged a lot. Definitely open source from NVIDIA is having a significant impact on how we think about our future workflows: NVTabular, HugeCTR, etc." They also have a graph neural network model, which is powering all of the substitutions in online grocery.

> Key Challenges in Recommender Systems

Hardware plays a vital role for recommenders at Walmart Global Tech. "Latency is something very important. We cannot build complex models and score them online. Usually, we'd use a simple regression—something that we could infer within 30-40 milliseconds. But then we had some deep learning models which showed plenty of promise. With GPUs you're able to seamlessly test and scale deep learning models in production, and that is really making a very big difference."

Another area which the team watches closely is in privacy and fairness. "We're always cognizant of it. that is an area that we take seriously." This is where interest in contextual bandits for recommenders comes into the picture, for reasons similar to those at The New York Times. Kannan explains, "We are seeing a lot of traction in exploration/exploitation because it's a well-understood problem. Contextual bandits and modeling of rewards in a rich way is something in which we are seeing very healthy payoffs. The exploit framework helps on revenue, but it also helps in presentation bias. We have shown on paper that it can be very useful for the business and also make our model better because it tries to unbias datasets."

To unpack this approach in more detail, this is a matter of using contextual bandits as building blocks, efficient units of learning strategically located within the workflow. "Traditionally, recommenders may show the top 10 candidates and users interact with only the top 10 candidates, and the bottom or 20 or 30 never get a chance – unless some items go out of stock. But if the models explore/exploit, we continuously do online learning where if something performs really well, but the variance is very high, we quickly show it more to make sure if it works well. The same goes if something is bad, but you're not sure – we show more of it to make sure that we can weed it out. We do this in a very principled way using a variant of Thompson sampling."

In other areas of research, the team is iterating on deep learned applications. "That is one big area we are working on – online inference using knowledge embeddings and deep learned models. The NVIDIA Triton Inference framework is so elegant that it seamlessly allows us to go between a CPU-based model and GPU-based model effortlessly. We use 'wide and deep nets' where, in some cases, we have a rich representation and the GPUs kick in, but when we don't have a deeper representation, the 'wide' network kicks in. if we had implemented this, it would have taken us a while, but that framework really accelerated implementation."

Looking ahead, Kannan cites two additional areas of technology to augment their recommenders. "One area that I think has a lot of promising research is in self-supervised learning. When you look at the site, we know someone viewed it. maybe it took 10 minutes for them to add the cart and 30 minutes to check out, and we know that evolution. The store data is 'This customer bought this item.' We really didn't know what made them buy it, we didn't know the order of items." Previously the team would need to correlate across omnichannel data, attempting to marry these two sources of data. "One comes as a basket of items, the other comes often with clear intent and how intent translates to an action." Self-supervised learning offers means for reconciling customer understanding across these data sources. In other words, sampling from one source to predict related aspects of the other source—as one might imagine, the optimal version of four blind men comparing their observations

about an elephant. Generative adversarial networks represent another way that unsupervised or semi-supervised pipelines may augment data sources and learning overall. "An area that we have been investing in for the last year is using GANs. We are getting some good results with adversarial networks, especially in the area of privacy and attack migration."

Even Oldridge, NVIDIA

Leads the NVIDIA Merlin™ engineering effort at NVIDIA

Even Oldridge leads the NVIDIA Merlin engineering team at NVIDIA. Merlin is an open source framework that provides support for building end-to-end GPU-accelerated recommender systems. Capabilities include support that spans from data preprocessing and feature engineering to training deep learning models and running inference in production.

Even describes, "Merlin is NVIDIA's play towards recommender systems. It started out as a focus on deep learning-based recommender systems, which is a passion of mine." Echoing key points of the advice from recsys experts in industry—namely, start simple—capabilities within Merlin track with customer needs throughout their project lifecycle. "With Merlin, [we] provide [a] framework so that you can start with the simple models and then build up over time to more complex. It is a gap in the field. It's currently very hard to make the transition from traditional machine learning to deep learning, and a lot of companies struggle with that stage and have to spin up a completely different pipeline to do the deep learning-based work and it's complex." It's also expensive, and as Monica Rogati, and other experts caution, those kinds of discontinuities in a project's roadmap can translate to risks.

Even adds, "We're really just trying to solve the challenge of how do teams build good recommender systems. How can we make that easier? How can we provide not necessarily the final solution, but a solution at each stage of the pipe. "Developing and deploying recommender systems is an incredibly complex task today. Machine learning in general is, but recsys even more so. That's shifting though, and we're very focused on Merlin as an easy-to-use, easy-to-deploy, performant framework."

One aspect of this approach is to keep in mind that every customer must have a roadmap; the requirements today stated aren't static.

"The work the HugeCTR team does is a great example of NVIDIA's focus on the upper echelon of recommender systems. The current project they're working on is a 100 terabyte model. That's bigger than any customer we know has, but the customers are going to get there soon. And when you get to that scale, you really have to be smart and think, and they're looking at what teams are doing and are publishing in terms of the open source space and trying to integrate that into the product, and they're also providing that guidance to other teams. Some teams adopt the technology just by integrating it into their own stack. At that level, you can, and I think the work that's gone on, a lot of it's been at that upper end and focused on that, and it makes sense. There's a lot of business there. The scale of companies working on recommenders, it varies dramatically. And the compute required at the upper end, it dwarfs the compute required at the lower end."

> Addressing the Important Hurdles

It's difficult to build machine learning models that will handle complex needs. As mentioned, building a model is not enough, and so far there have been portions of the typical machine learning workflow that received less attention. Correcting this gap is especially important for a thriving open source ecosystem. "How do you get those models into production? And that is a huge barrier for

a lot of companies. It's not easy, especially on the deep learning side, but even a basic model for collaborative filtering, getting something into production and then properly monitoring it and making sure that it's not going off the rails and making sure that you're tracking all the things you should be tracking. It's immensely complex. And so the second thing that the [Merlin] team works on is making it easy to deploy into production." De-risking these aspects of recommender practice is the other priority for the Merlin team.

> Initiating Recommender Projects

For people who are just starting their recsys journey, Even advises, "I think that the key is to start simple and iterate. There's a tendency to want to build the latest, greatest deep learning model, I think, amongst data scientists universally. And starting out with something more straightforward and simple to make sure that the signals are there. Good data is always better than a good model, in my experience, and I think that's a lesson that data scientists often learn the hard way. But I feel like it's a pretty universal state, and keeping your data maintained and clean and then doing feature discovery often will lead to way more benefit than starting from an 'I'm going to try this latest model' perspective."

Earlier in his career, Even built recommenders in production for a popular online dating system. He urges practitioners to get a process in place so that they understand the data being collected, the metrics being measured, the deployment environment, and the objective function for the use case—well in advance. "Here's the model, and here's how we're going to iterate on it, and the output of that is going to be a model of this type." Then work closely with the teams handling production to understand their process, their constraints, and so on.

SUMMARY

The interviews illustrate how there's much common ground among different recommender systems practices in industry, even when we look across different business verticals: retail, news, social media, gaming, and so on.

First and foremost, open source has become an essential component for recommender systems in industry. For de-risking projects, it's critical that tools within an open source ecosystem are interoperable, especially when considering projects over their full lifecycle. Tooling needs to be flexible enough to support experimentation and exploration, while admitting new technologies that haven't yet emerged.

Another given is the matter of data. Notions of "big data" emerged in the mid-2000s. Since then, it's generally recognized about machine learning in production that having more and better-quality data tends to result in more effective models. Certainly this is the case for recommender systems as well. We cannot depend solely on better algorithms to improve models. Instead, we also need good training data, good customer data for inference, good feedback and instrumentation for evaluating metrics, and so on. Priorities for effective data preparation will likely be a perpetual requirement. Note that data rates will grow as a recommender use case gains success. It's also vital to keep learning from the data as you refine your metrics for success. Think of reframing your data science staff as "metrics engineers"—for the sake of thoroughly understanding recommender use cases in production.

The following takeaways summarize other key points that have been emphasized within these expert interviews.

Takeaway #1: Start Simple

The consistent advice from experts for starting a recsys practice is to start simple. Ideate about what's needed. Instead of rushing to implement a complex machine learning model that's trending in industry reports, try mocking with a simple rule-based baseline first. Follow a process to test, measure, iterate. Then your solution can progress from simple to complex, with less risk.

Pete Warden has written about a related practice—originating at JetPac and carried through into Google—called "Wizard of Oz-ing" where teams first mock the machine learning components of a proposed product performed interactively in front of a live audience.

Meanwhile, begin by using well-known, proven technologies. Prioritize ease of integration and keep your ratio of efforts to results as low as possible.

Takeaway #2: Understand the Objective Function

Determine your metrics for success, and thoroughly understand the objective function for the use case at hand. This is the single, most repeated point of advice.

Takeaway #3: Extending into the Business Model

Looking back at the Netflix Prize (2006–2009), that recommender challenge was formulated as a point solution. In other words, the recommender was considered responsible for supplying a list of ranked items. There were inputs and outputs, with a "black box" solution in between.

Since that 2009 period, we've seen how recommender systems in production have changed business models. For example, grocery stores are generally considered to have thin margins. Early attempts to build grocery-delivery business models struggled. With the introduction of recommenders, machine learning could be applied, not just at the point of contact with the customer, but deeper into the business process.

Recommenders now address a much broader scope of personalization throughout the user experience. They develop contingency plans and help optimize other aspects of business. In the case of The New York Times, machine learning solutions support the economic strength of the business, with recommenders optimizing for building readers' trust and loyalty. This is done in a way that scales editorial judgment rather than displacing it.

Within the period of merely a few years, the use of machine learning has shifted from a question mark to a core business strategy. The evolution of hardware over that period, concurrent with the evolution of much more complex, capable machine learning technologies, changed that perception in business cultures.

Takeaway #4: Fast Inference and Hardware Acceleration

Historically, open source projects and vendor offerings focused more on the early stages of a typical machine learning workflow, such as ETL and data preparation. The later stages, such as model serving and real-time inference—which are closest to the customer experience—tend to get relegated to ad-hoc code that's part of the end-use application. Even so, many production use cases for recommender systems have fast inference requirements, typically under 100 milliseconds.

While there are many options and interesting technologies that could be leveraged in recommenders, the time required for inference is a gating function. If quality recommendations cannot be produced within the time window, then simpler technologies must be used instead. For example, even at sophisticated, well-resourced practices, up until about two years ago, most of the machine learning

models were relatively simple linear models. However, more recently, deep learning models have been introduced. The game changer was hardware allowing for fast inference. The net result is that hardware enables more sophisticated AI adoption overall, which ties into the earlier takeaway about recommenders extending deeper into business models and operations.

As the field of machine learning continues to evolve, there will undoubtedly be more sophisticated approaches emerging. For instance, GANs, self-supervised learning, and reinforcement learning each promise interesting applications for recommender systems, although much of this work is still at a research stage. These specific examples also increase the amount of computing resources required.

One question is: Will more advanced technologies be performant enough to run within milliseconds? Does that constraint preclude recommenders in production from adopting the latest innovations? Instead, orchestrated clusters providing large memory spaces and hardware acceleration will become an enabling factor for more advanced AI applications.

Takeaway #5: Key Components

Hot topics related to recommender practices in industry include using feature stores (introduced a few years ago), making use of graph neural networks, and model distillation.

Takeaway #6: Plan for a Roadmap

Plan ahead for system upgrades. In terms of components, such as simple linear models getting replaced by more complex deep learning models, this may require changes to your operational environment. In a very different sense, as a recommender practice helps power the economic vitality of a business, there will likely be reasons to grow the practice. Data rates will probably grow. The recommenders over time will extend deeper into the business process. Or other business units may require additional recommender use cases.

Plan for a roadmap of system upgrades and growing use cases. Where possible, leverage technologies that provide a roadmap for scale, not simply for the scale requirements today but for what your successful business will need a few years from now as well.

Out of the billions of people in the world, and the many moments they spend engaged online—browsing, shopping, learning, catching up with friends—each moment, each event, each session is another opportunity for recommenders to help make informed decisions that are easier, faster, and more personalized for each individual. This implies billions of people interacting with trillions of things online.

LEARN MORE

To take a deeper dive into NVIDIA Merlin, visit: developer.nvidia.com/nvidia-merlin

NVIDIA

IN-DEPTH INTERVIEWS WITH LEADS AND EXPERTS

In the first half of this report, we summarized trends and excerpted interviews with leads and experts in their domains. In this section we have provided a few of the in-depth interviews. In certain cases, the interviews have been edited for readability with approval of the interviewees.

Q: What is the size of your data that you've used for building recommenders?

If we use a generous definition for recommenders, the range of the data I've used varied from zero data points to matching hundreds of million users to as many items. The upper end of this range gets most of the attention as it can address the long tail and make very personalized recommendations that feel like magic (or creepy, depending on your perspective). Bigger and more diverse data is also where the interesting algorithms that data scientists gravitate towards shine.

Q: How do you choose the appropriate technologies and frameworks to support your work?

I'm very pragmatic about recommending technologies and frameworks. There are a few criteria I keep in mind, in addition to the likelihood of solving the problem: ease of integration, maturity and preserving optionality.

One of the most important qualities is ease of integration with a company's current workflow, and keeping the ratio of effort to results as small as possible. I remember a company wanting to sell my team at LinkedIn a particular technology, us being very motivated to try it as we were hitting issues at scale, but having no way to test it unless we sunk a considerable amount of effort and engineering time into it. That's a risk you can take with well known technologies that have been proven again and again -- but not with a new entrant. As an aside, " Use boring technology" is the "Eat your veggies" of data science -- people know it's a good idea in theory, but that latest-and-greatest fun tech is extremely tempting for engineers and data scientists. While 'boring' (mature) technology is particularly important at scale, there's always room to try the latest tools on smaller problems off the critical path. Explore and exploit tradeoffs are not just a recommender system issue -- you can think of trying new tools as an investment in innovation, or simply a morale, recruiting, retention boost for your data team.

In the past few years, there's been an explosion of data tools and frameworks, and I'm happy to see that many of them do have easy integration points that don't require significant changes to the workflow until the tool is proven and trusted in a particular environment. My go-to question for people building these tools is 'What's your Trojan Horse'? What is the *one* problem you could solve with low or no effort on the practitioner's part to gain adoption and trust?

Last but not least, preserving optionality is important in the early days of adopting a framework or technology. What happens if it turns out it's not the most appropriate technology after all, or our needs change? This is not just about lock-in, it's about de-risking technologies early in the process and having a backup plan. The best tools include possible exit paths in their marketing and provide data export tools in addition to easy integration points.

Q: How important is interoperability with open source?

Very important -- especially in the context of choosing appropriate technologies and frameworks. 'Ease of integration' comes for free when you can plug-and-play into various open source technologies. Similarly, knowing that replacing a piece of your (hopefully) modular technology puzzle is minimally disruptive to the rest of the system goes a long way towards the 'preserving optionality' criterion above.

MONICA ROGATI

AI and Data Science Advisor

Creator of the First Machine Learning Model for LinkedIn's "People You May Know" Feature

Monica Rogati is a technical AI and data science advisor to companies ranging from 5 to 8,000 employees and spanning several industries. Before becoming an independent advisor, she was the VP of Data at Jawbone where she built and led a team of top data scientists and engineers. Prior to Jawbone, Monica was an early member of the LinkedIn data science team where she developed and improved LinkedIn's key data products including matching jobs to LinkedIn members, discovering people you may know, and recommending professional groups. Her background includes applied machine learning (Computer Science Ph.D. from Carnegie Mellon), NLP and wearables. Monica's "AI Pyramid of Needs" diagram has been serving as a shield to misunderstood data scientists everywhere.

Q: How do you evaluate models?

I'm going to assume that if we got to this point, we have a vetted problem where the training data has been obtained and compiled in an ethical manner, and that the system is not harmful when maximizing the given objective function. This is much, much harder than it seems; and 'good intentions' are far from a satisfactory solution. You'd probably need to do some research as expert-endorsed best practices have evolved over the past few years.

Back to the original question of evaluating models, I usually recommend three steps: baseline, lenient offline filter, A/B testing with slow ramp up.

First step is analyzing the data and implementing a reasonable but ridiculously simple rule-based baseline. For recommendations, this could just be 'the most popular items over the past 3 months'. Or, if additional information is available, adding another property to the rule: 'the most popular items over the past 3 months in the person's country/state'. For a classification problem with relatively few positive examples, it could be 'Just say NO all the time' (and this is why claims of 99% accuracy for relatively rare events are misleading, albeit technically true) .

If there's already a model in place and you skipped this step, try it anyway. Simple baselines are easy to debug, robust, fast, and they should be preferred if it turns out they're almost as good as your current model.

The 'lenient offline filter' is a twist on the traditional offline model evaluation. The goal here is to check that the model is not a complete disaster that could result in huge disruptions or unexpected harm and looks 'promising' in a very loose sense of the word. The goal is *not* necessarily to beat the current system (although if you started with the baseline above, it should.)

Finally, the true evaluation is A/B testing in production with a very slow ramp up. We're watching for operational issues, missing features, unexpected harm and generally problems that might not appear offline or at a small scale. After the model is at scale, especially for recommender systems, often there needs to be a 'burn in' period to allow any novelty effects to wear off.

Q: What have been some challenges with recommenders that you had to address?

One of the biggest challenges is designing the right metric or objective function. Data scientists will eventually become metric engineers -- it's one of the data tasks that are hardest, if not impossible, to automate.

Q: If a team lead was just starting out and currently evaluating building, deploying, and optimizing recommenders, --- what advice would you relay?

Think carefully about the metric you're trying to maximize (your objective function) -- often, it's an easier-to-measure proxy for what you're *actually* trying to achieve. Will it still be a good proxy after you influenced it with your system? Is there a way to game it, willingly or inadvertently due to incentive alignment? How hard is the problem and what's the upper limit of what you'd expect as far as performance goes -- for example, what's the inter-annotator agreement on your training data?

There's also something I've learned from Salim Roukos when interning at IBM Research: before starting to work on a machine learning system, spend an hour labeling data and doing the algorithm's job. Be the algorithm. You'll find countless data quality issues, identify a set of situations when the question is ill-defined, or when you should abstain from making a recommendation.

XIANGTING KONG
Expert Engineer
Tencent

Q: What is your role at Tencent?

I am an expert engineer at Tencent and am responsible for the design and development of the advertising recommendation system. I am also the lead of the Tencent Advertising and Deep Learning Platform. Our platform supports the machine learning model optimization, training, and inference in a variety of business scenarios from advertising and fintech to networking data mining.

Q: What does your team at Tencent work on?

Our team mainly develops machine learning platforms and we are responsible for feature engineering, model training and online inference. We are working on implementing a new generation of high-performance distributed training system for advertising recommendation based on GPU from 0 to 1.

Q: How does your work and your team's work on recommenders relate to Tencent's overall business?

Our advertising recommendation training platform covers the entire Tencent traffic business. Tencent advertising recommendations are widely used in services such as WeChat, Moments, QQ, Tencent Games, Tencent Video, Tencent News and so on. Tencent advertising revenue is in the hundreds of millions. The accuracy of our advertising recommendation helps increase advertising revenue.

Q: Is your team a relatively new team? Why did Tencent decide to invest in recommenders?

Our team has been established for years. The advertising business is a relatively important business inside Tencent and the recommendation system is used to increase the overall advertising revenue.

Q: What kind of recommender systems does your team focus on?

The main focus of our team is the advertising recommendation system, responsible for the optimization of the advertising training platform. Tencent advertising recommendations system consists of parts including offline feature engineering, training platform, online inference system, online feature engineering and play platform. Advertising recommendations is a process of gradual filtering. Sorting stages include recall, pre-ranking, and ranking. Each stage has different requirements. The rapid investigation and iteration of the model puts forth higher requirements for training performance.

Q: How does your team conduct training?

We organize some technology sharing every one or two weeks.

Q: How does your team evaluate your recommender systems? Fine tune?

Through our recommendation system, we optimize the algorithm strategy, add more samples and features, and assess whether it can drive the increase of income. The accuracy of the advertisement recommendation can be improved by training more sample data, by adding more sample features. But this leads to longer training time and effects the update frequency of the model. In order to ensure that the model updates will not be derailed, the training performance

of the model needs to be continuously improved. After the training model performance is improved, more data can be trained to improve the accuracy of the model, thereby increasing the advertising revenue.

Q: How do you optimize your recommender systems? For example, it is our understanding that Tencent uses HugeCTR for embeddings optimization. How has this helped you optimize your workflow?

HugeCTR, as a recommendation training framework, is integrated into the advertising recommendation training system to make the update frequency of model training faster, and more samples can be trained to improve online effects.

Q: How do you choose the appropriate technique, package, method, frameworks to support your work?

The technology or framework we choose must be compatible with the community ecosystem, so that we can do better follow-up upgrades.

Q: How do you address scaling your models?

Using a larger model is conducive to learning more features, thereby improving the accuracy of the model.

Q: What is a recent success for the team?

In our training framework, a data-parallel distributed solution has been developed.

Q: Have you recently integrated specific methods into your recommender workflow?

We recently integrated the CSR [Compressed Sparse Row] pipeline into our ad recommendation training system. CSR type training data is generated so that the data can be directly read on the GPU for training. Through our optimization of the data processing pipeline, the CPU load is greatly reduced and the GPU utilization is greatly improved.

Q: If a team lead was just starting out and currently evaluating building, deploying, and optimizing recommenders for their company, --- what advice would you relay to help them accelerate or streamline their recommender workflows?

Choose a mature technical framework and be compatible with the community ecosystem to facilitate subsequent system upgrades.

CHRIS WIGGINS

Chief Data Scientist
The New York Times

Q: What is your current role at The New York Times?

Chief Data Scientist, leading a team to develop and deploy machine learning solutions to newsroom and business problems.

Q: How does your work and your team's work on recommenders relate to the New York Times overall business?

The Times has a goal of reaching 10 million paid subscribers by 2025. Subscribers are interested in The Times' editorial judgment, so recommendation

means optimizing for trust and loyalty, and ensuring that we do so in a way that scales editorial judgment rather than replacing it. Right now recommendations appear on a variety of news surfaces, including "More In", "Editors Picks", "Smarter Living", the international home page, and "Most Popular". We also power recommendations in the Cooking app, the "For You" tab in the mobile app, and the "Your Weekly Edition" newsletter.

Q: Is your team a relatively new team? Why did the New York Times decide to invest in recommenders?

Yes and no. Personalization has evolved over my time at NYT, since 2013, with plenty of "dynamic reteaming", so there's been a continues lineage but with changing line-up and roles, like a British rock bands poster except driven by changing infrastructure capabilities and product goals rather than whatever excitement lead Jimmy Page to leave the Yardbirds. In June, we had a re-sort within data science in which Dr. Anna Coenen, who had previously led personalization of the homepage, is now leading Algorithmic Recommendations team (Dr. Anne Bauer, previously leading the team, is now leading a team for ML for onsite marketing). Recommendations have been part of the digital strategy since well before I arrived (in 2013). At present it's seen as part of our user experience and business goals.

Q: How do you choose the appropriate technique, package, method, frameworks to support your work?

As a strategy, Kreps Law. I have a keyboard alias for it.

As a tactic, it depends a lot on the goals of our product and newsroom partners, who are very thoughtful about the "x" (the context), the "a" (what possible actions or articles or assets are under consideration to recommend) as well as the "y" (what outcome we're trying to optimize).

Q: How do you address scaling your models?

Our infrastructure is deployed via Google Cloud Services. Data ingest is via a proprietary event tracking service described in a 2018 Open blog post. Requests to our recommendation APIs to power recommendation while respecting user privacy are via an internal service called Samizdat, described in a 2021 Open blog post.

Q: What have been some recent successes or challenges that were successfully addressed?

One success was moving from predictive methods to prescriptive methods, e.g., different contextual bandits (described in a 2019 Open blog post) which can be deployed for the separate surfaces and pools of content we power. There's not one structureless feed, so part of the scaling challenge is to iterate on new algorithms and new surfaces, where clear thought about the data, including the "shelf life" of news as well as what is being optimized for, can make a huge difference. Another success was deploying recommendations on the Cooking app, which has been a very successful product as a "standalone" product and is more and more integrating with NYT infrastructure, enabling lots of new product development experiments. The content is very different from "news", and there's been lots of creative partnerships between data science and product to make sure we're capturing and empowering an editorial vision there.

Q: Are there any recent recommender system papers on specific techniques or methods that you find intriguing that you are considering integrating into the workflow? Or have you integrated specific methods?

I mean, yeah but not that we're sharing publicly yet. Keep an eye on open.nytimes.com.

Q: If a team lead was just starting out and currently evaluating building, deploying, and optimizing recommenders for their company, --- what advice would you relay to help them accelerate or streamline their recommender workflows?

I tend to lean on the shoulders of other people's advice, e.g., Kreps Law, and of course Monica's infographic. Some NYT-specific success has come from co-creating by close pairing between data scientists and software engineers, who sit (or sat, when we were in office) side-by-side, coding in a mixture of Python (especially the data scientist) and Go (especially our software engineering partners) to drive methods that are performant both in statistics and in SLAs.

JUN HUANG

Senior Technical Expert of Meituan

Q: What is Meituan? How many users and merchants are using Meituan? How many transactions happen on the Meituan technology platform?

As the largest life service company in China, Meituan's business covers more than 200 service categories such as catering, distribution, car hailing, bike-sharing, hotel, tourism and movie ticketing, and covers more than 2,800 cities and counties in China. Meituan has 630 million active users and 7.7 million active businesses every year. Each user has 32.8 transactions per year.

Mission: Help people eat better, live better.

Q: What is your role at Meituan?

I am a senior technical expert of Meituan, mainly responsible for the training framework team of Meituan Machine Learning Platform. Our training framework covers a large number of deep learning scenes in Meituan, including: recommendation system, NLP, CV, ASR, automatic driving, etc.

Q: What does your team at Meituan work on?

Our team developed a stable and high-performance distributed deep learning training framework. Our system can be deployed in large-scale CPU/GPU clusters, and supports failover and auto scaling. In the recommender system scenario, we support distributed training with ~100 billion sparse parameters and ~100 billion samples, and also support online learning. In NLP scenario, we support distributed training with ~10 billion parameters on hundreds of GPUs. Recently, we have designed the next generation of recommender training system based on NVIDIA A100, which greatly improves the training efficiency and the model complexity.

Q: How does your work and your team's work on recommenders relate to Meituan's overall business?

Our training framework covers the model training of search, recommendation and advertising scenarios , which are the overall traffic business of Meituan. We helped Meituan accelerate the growth and realization of flow.

Q: Is your team a relatively new team? Why did Meituan decide to invest in recommenders?

Our team has been established for many years and is the infrastructure team of Meituan.

Search, recommendation and advertising are the most important business flows in Meituan, which will bring more user growth and business realization.

Q: What kind of recommender systems does your team focus on?

For the recommendation system, our team mainly focuses on the optimization of model training. We designed an efficient data flow system, which is convenient for reading offline and online features flexibly and efficiently. We optimize the system from the perspective of scaling out, and make the system expand nearly linearly. We also optimize the system from the perspective of scaling up, and fully utilize the hardware resources. We also do jointly optimization for training performance and accuracy from the system and algorithm perspectives. We continue to pay attention to SOTA of new models and new hardware, and bring valuable technologies to Meituan's business.

Q: As Meituan has 630 million users and many interactions per user, how does your team train job tasks? How frequently do you train?

Usually, a small number of samples are used to try the algorithm strategy first. If the verification is effective, a large number of samples are used to experiment the above strategy, hoping to get better results.

Every day, there are various experiments, but for online models, the training frequency varies from one day to one week.

Q: How does your team evaluate recommendation systems? Fine tune?

In our recommender system, we generate a series of models by experimenting with various model structures, algorithm strategies and sample features, and judge whether these models can bring the improvement of business indicators through offline and online evaluation. Usually, training larger samples and more complex models will improve business indicators, but this will consume more resources and reduce the number of experiments. We will balance this, but if we can greatly improve performance, we usually prefer to train more samples and more complex models.

Q.How do you optimize your recommender systems?

At first, we optimized the training framework based on CPU architecture, but as our models became more and more complex, it was difficult to optimize the training framework deeply.

Now, we are working on integrating NVIDIA HugeCTR into our training system based on A100 GPUs. A single server with 8x A100 GPUs can replace hundreds of workers in the CPU based training system.

The cost is also greatly reduced. This is a preliminary optimization result, and there is still much room to optimize in the future.

Q: How do you choose the appropriate technique, package, method, or frameworks to support your work?

This technology needs to be advanced, open and ecological, so that we can fulfill our internal needs better based on it.

Q: How important is open source technology and interoperability to your team's work?

Very important. Our team is currently building our system mainly based on open source technology. At the same time, we are very happy to give back our work to the open source community.

Q: Recently Meituan reported an average increase of 32x more transactions per user for the trailing 12 months of Q2 2021. How does your team handle scaling your models?

We used more samples and more complex models to express our business model, which greatly improved our business effect.

Q: What is a recent success for the team?

Based on A100+HugeCTR, the training framework for the next generation of recommender system is developed, and achieves the preliminary results.

Q: Have you recently integrated specific methods, techniques, plugins, libraries, or packages into your recommendation systems workflow?

Made the whole data flow pipeline, from the remote distributed file system to the local CPU memory, then to the GPU memory, so that the computation and IO can be overlapped. Embedding Layer and Dense Layer are also pipelined, which makes embedding layer no longer be the bottleneck of the whole system.

Q: If a team lead was just starting a team and evaluating building, deploying, and optimizing recommendation systems for their company, ---what advice would you relay to help them accelerate or streamline their recommender systems workflows?

Fully understand the current company's infrastructure and business status, and design systems and processes based on it. When choosing the technology stack and framework, consider the maturity, community ecology, extensibility and integration friendliness of each system.

FELIPE CONTRATRES

Tribe Leader of Personalization
Magalu (Magazine Luiza)

Q: What is your current role at Magalu? What are you responsible for?

I'm currently responsible for the Magalu Personalization Team as a Tribe Leader. My responsibilities are to ensure that we deliver quality and useful software to production, aligned with Magalu's business objectives.

Q: What does your team at Magalu work on? What is your team responsible for?

My team works on building our Recommendation and Search platforms. In my team I have developers and data scientists working to evolve our systems. My Recommendation Team works on creating new functionalities for our platform and new recommendations models.

Q: How does your work and your team's work on recommenders relate to Magalu's overall business?

Recommendations are responsible for a considerable amount of Magalu's revenue, participate significantly in the user experience and it also has a lot of impact in several of Magalu's KPIs.

We provide recommendations for many of Magalu's channels (desktop website, mobile website, app, MagazineVocê and in all of our +1400 physical stores). We also enable our Advertisements Team (Magalu Ads) to publish recommended ad algorithms in our platform.

Our platform recommends millions of products and services for 32 million active monthly users.

Q: Is your team a relatively new team? Why did Magalu decide to invest in recommenders?

Magalu has had the recommendations team since 2013. At that time, we had a third party company providing our recommendations. This being a crucial point for our Marketplace Platform made us decide to create an internal team to have our own Recommender System.

Q: What kind of recommender systems does your team focus on?

My team focuses on product recommender systems. We built a Recommendation Platform that allows other teams to collaborate in creating new recommendations models. We are responsible for maintaining the Platform and creating new features to be used by our Business Team.

Q: How does your team handle pre-processing (ETL) and feature engineering?

Our Data Platform Team created an internal and amazing Data Platform that we use to process all of our data ingestions and pre-processing. We also have an internal Machine Learning Platform that we use to store our features and train our models.

Q: How does your team evaluate your recommender systems? Fine tune?

We use offline and online evaluations for our new models. We built an A/B test feature in our Recommendation Platform that allows anyone to put two different models online and compare their efficiency in production.

Q: How do you optimize your recommender systems?

By testing them online with our A/B test feature. We create new versions of our models and put them online with an A/B test.

Q: How do you choose the appropriate technique, package, method, frameworks to support your work?

We start by looking into the Recommer Systems literature to understand if there is any previous work on the issue we are dealing with. To decide which package or framework we'll use, we always check how engaged the community is and how well established the framework is.

Q: How do you address scaling your models?

Here at Magalu, we have an amazing Machine Learning Platform that handles the complexity for us. This platform allows us to train our models and expose them in a scalable way, abstracting the complexity related to infrastructure.

Q: What have been some recent successes or challenges that were successfully addressed?

We successfully achieved our goal of building a flexible and scalable Recommendation Platform, in which other teams can contribute by creating new specialized models and customizing the user experience in an easy way.

Q: Are there any recent recommender system papers on specific techniques or methods that you find intriguing that you are considering integrating into the workflow? Or have you recently integrated specific methods?

We are considering testing the technique proposed by Gabriel de Souza P. Moreira et al in the recent paper "Transformers with multi-modal features and post-fusion context for e-commerce session-based recommendation".

Q: Are you or your team evaluating, or using, session-based recommenders?

We currently have an heuristic session-based recommender that customizes the intra-session user experience in real time and we are evaluating the proposed method of the paper cited above ("Transformers with multi-modal features and post-fusion context for e-commerce session-based recommendation") to improve our customer experience.

Q: If a team lead was just starting out and currently evaluating building, deploying, and optimizing recommenders for their company, --- what advice would you relay to help them accelerate or streamline their recommender workflows?

I have some advice:

- Start with simple approaches: many problems are easily handled by simple techniques that reduce your overall system complexity.
- When more advanced techniques are necessary, choose mature packages or frameworks to integrate into your workflow.
- Try to put your model online as fast as possible to test it with real customers instead of trying to pre-optimize excessively offline.

VINNY DEGENOVA

Associate Director of Data Science - Search and Recommendations at Wayfair

Q: What is Wayfair?

Wayfair is an American e-commerce company specializing in furniture and home goods, with a leading presence in the US, Canada, UK, and Germany. We currently have a catalog of approximately 22 million products and an annual revenue of more than $14.8 billion.

Q: What is your role at Wayfair?

I've held a few roles at Wayfair spanning the recommendations space, specializing in personalized recommendations. Over the past few years, I've built out our team that specializes in personalized product recommendations. I am currently an Associate Director on the Search & Recommendations team, leading our Content Recommendations Data Science team, focused on recommending the right content (sale events, videos, inspirational articles, email content, etc.) to the right customer at the perfect time to facilitate their journey both on and off-site. Our goal is to create a unique and personalized shopping experience for each customer.

Q: What is your team at Wayfair responsible for?

My team is responsible for developing the recommendation algorithms that power content recommendations across the Wayfair website, app, marketing emails, push notifications, and advertisements on third-party websites. We partner closely with our engineering team to productionize and scale these systems to serve billions of recommendations per day, and work very closely with our analytics partners to quantify their impact and assess the opportunities we should tackle next.

Q: How does your work and your team's work on recommendations relate to Wayfair's overall business?

The work we do on personalization has a profound impact on the customer experience, making it easier for customers to find the products they love, and has a direct measurable impact on Wayfair's business. We leverage A/B testing to explicitly measure the performance of new recommendation models, as well as updates to recommender systems we deploy on-site in terms of sales and user engagement.

Q: Is your team a relatively new team? Why did Wayfair decide to invest in recommendations?

The recommendations team at Wayfair predates me at the company! I've been here for the past 4 and a half years and it's clear that Wayfair, as an e-commerce company, has seen the value in investing in high-quality recommendations for quite some time. While the systems we use to recommend products and content more generally have evolved over time from relatively simple aggregations of co-clicks and collaborative filtering models to more robust deep learning, graph-based, and reinforcement learning solutions, it's been an integral part of Wayfair's business model to have a state-of-the-art tech stack when thinking through what to display to customers.

Q: What kind of recommendation systems does your team work on?

My team focuses on developing recommender systems that span multiple types of content, and we take a few different approaches to generate recommendations for different use cases. For example, when generating personalized recommendations for sale events currently happening on-site, we use a combination of reinforcement learning techniques to understand which sale events are the most popular and develop custom heuristics based on a customer's previous shopping behavior to tailor the list of sale events they see in their email. For generating personalized product recommendations, our teams have invested in building out deep learning techniques to rank products on the Wayfair website in real-time based on what we think customers are most likely to engage with. For more details on the models we've built, check out some of our posts on the Wayfair tech blog here.

Q: As Wayfair customers have shown a preference to "shop for one item, one room, and one project" at a time, how does your team handle providing relevant recommendations?

One of the most important pieces to master when developing a recommender system is a robust understanding of the candidates you're recommending. In the case of product recommendations, this is typically owned by partner teams on data science at Wayfair that automate processes to algorithmically tag products affiliated with a particular style, for example. For other types of recommendations, we build dedicated feature derivation pipelines that can extract meaningful information from imagery, text, and other sources that we can use in conjunction with a customer's browsing history to recommend the most relevant content across each touchpoint they have with Wayfair. This gives us a deep understanding of the items we can recommend to customers. In parallel, we analyze how customers interact with the Wayfair website to understand what they're looking to purchase, and we aim to leverage recommendations to remove friction from the customer buying cycle, showing them the products and content most relevant to their needs.

Q: How does your team handle preprocessing (ETL) and feature engineering?

We partner closely with our Machine Learning Platforms team to think about feature reusability. While there will always be niche features for specific use cases, when it comes to recommendations, typically many of the features we're using are quite similar across use cases. Wayfair has heavily invested in building out a platform that allows us to create features once that refresh regularly and can be reused across multiple projects, multiple teams, and are accessible in offline formats (typically in GBQ tables), or are accessible online for models that run inference in real-time.

Q: How does your team handle training? How frequently does your team train?

Training frequency is definitely something that I see as fairly use case-specific. As a general rule, we leverage Airflow for model orchestration and scheduling and distribute our training jobs across resources using Google Cloud. For more predictive modeling, we schedule training jobs to execute anywhere between daily and weekly, depending on the task.

For more online learning systems, we use batch updates at a much higher frequency, targeting intra-day updates that allow us to respond to changes in customer behavior as quickly as possible.

Q: With over 31 million active customers, 22 million products on the platform, and serving billions of recommendations each day, how does your team handle scaling your recommendation systems?

Scalability needs to be baked into model development from the very beginning. With a catalog of this size and a customer base at the scale we operate at, there isn't room for inefficiencies in model training, ETL pipelines, or model inference. Every model we work on is tested and evaluated, not only from the perspective of showing relevant items to customers but also in terms of its scalability and maintainability. Having clear visibility into things like how long it takes to generate recommendations and how long it takes to train your model are part of the reason why we partner closely with engineering teams to flag and identify inefficiencies before models are live in production.

Q: How do you and your team evaluate your recommendation systems? fine tune?

We've developed an in-house package that allows our teams to evaluate and tune product recommendation systems in an offline fashion before going to an online A/B test. We backtest our models on historical data, assessing how well new models would have performed relative to the models that are used on site today. Generally, across all search and recommendation initiatives at Wayfair, we backtest and rely heavily on guardrail metrics (e.g. ensuring we're not recommending products with no reviews) and heuristics to ensure that new models we develop are improving the customer experience.

Q: When you and your team review the latest research, how do you and your team choose the appropriate technique, package, method, libraries, or frameworks to support your work?

I like to describe the work we do on the Data Science team at Wayfair as 'Applied Research'. That is, research applied to a specific business problem. When there isn't a documented solution for a problem we're trying to solve, we also have the flexibility to go to the whiteboard ourselves to think through an optimal approach. When we're looking for inspiration, or reading through the latest papers coming out of leading conferences, we're typically looking for techniques that can be applied and adapted to a Wayfair-specific domain. The framework or library we use isn't quite as important to me as is the value we're looking to drive for our customers, and making sure that any solution we're building is both maintainable and scalable as our catalog and customer base continues to grow.

Q: What is a recent example of integrating a specific method, package, library, or framework into your recommender workflow?

We've recently wrapped up a large migration from our own on-premise datacenters to leveraging the Google Cloud as our preferred platform for all data warehousing, model training, ETL pipelines, and model serving. It's been a highly cross-functional effort across all engineering teams here at Wayfair, but our Data Science teams are already starting to reap the benefits of a more efficient and scalable cluster to develop, train, and deploy models.

Q: If a team lead was just starting a team and evaluating building, deploying, and optimizing recommendation systems for their company.....what advice would you relay to help them accelerate or streamline their recommender systems workflows?

Start simple. Oftentimes, there is a significant amount of headroom in leveraging relatively simple models, or simple implementations that capture 'low hanging fruit' before moving on to more nuanced and complex solutions.

Don't reinvent the wheel. While there are new types of recommender systems being researched and published regularly, the field of recommender systems has been around for a while, which means there are plenty of open-source resources to leverage to accelerate standing up your own models. There's no need to build everything from scratch, leverage as much as you can from the community around you.

Develop with scale in mind, or else. If you don't take into account scalability from the inception of a new model, it will fail you when you move it into a production setting - the only question is when. It's wise to start your development process with a strong sense of how your model will scale over the course of weeks, months, or even years to come!

KANNAN ACHAN

Personalization and Recommendations Team lead

Walmart Global Tech

Q: What is your current role at Walmart Global Tech? What are you responsible for?

I lead the Personalization and Recommendations Team at Walmart Global Tech.

Q: What does your team at Walmart Global Tech work on? What is your team responsible for?

The team is responsible for personalizing the customer journey. Typically in recommended systems, it's a single recommendation carousel. It's basically a single module in a page. But the charter of this team is to look at whole page personalization. Everything comes to the personalization team in one shot.

Q: How does your work and your team's work on recommenders relate to Walmart Global Tech's overall business?

The personalization system has over 50 touch points on the customer journey, all the way from emails to discovery on home page, item pages, post transactions, marketing, even voice commerce. We do whole page personalization, but just doing the page alone is not enough. We think of it as a tensor in a way, and we have to personalize the whole session as well. We have to know who the customer is, number one. When the customer is on the site, we should know why they are here, allow them to discover what they have. That is where in the discovery funnel the recommendations really come into play, then help with conversions. More importantly, when you look at this journey, we also work on models which bring customers back to re-engage.

Q: Is your team a relatively new team? Why did Walmart Global Tech decide to invest in recommenders?

When I joined, initially one of our first projects was to build recommenders. Very soon afterward we're asked to get a unified view of customer understanding.

We had Walmart online, Walmart stores, Sam's Club, and we did see a lot of customers go shop in all these properties. The goal was to create a unified customer ID and get a good understanding of how customers shop. That was six, seven years back. That's when the company decided to invest in personalization and we began to personalize a few modules in the home page using that rich understanding of customers.

Q: What kind of recommender systems does your team focus on?

If you think of the space of recommenders, a traditional view of recommenders is a matrix with users and ratings – i.e., collaborative filters. If you look at discovery in general, we have to provide something that is more traditional such as "People who viewed this also viewed..." However, the moment an item goes to cart, we also have to recommend complementary items. What complicates omni-channel data is that there's also an intent. A customer could be shopping for groceries, or they could be making a one-off purchase like a television set.

Many customers come to Walmart for the grocery. The moment we predict the intent to be someone doing their grocery shopping, we quickly pivot and help them complete their basket as soon as possible. Not only do we look at basket size, we also have another metric to optimize how quickly we can help close this transaction. A more recent invention from the team is to predict the basket: with one click, the whole basket goes into the customers cart and then to checkout.

Q: How do you choose the appropriate technique, package, method, frameworks to support your work?

For something offline, data scientists generally have a lot of flexibility in choosing the models, the choice to build a proof of concept, etc.

But there is significant bearing on what can run online, because there's a 40 to 50 millisecond budget. A lot of times, that shapes the kind of technology we select. Sometimes we are even comfortable using simple logistic regression or gradient boosting decision trees because they really worked, they were predictably explainable, and we could control the latencies in a meaningful way.

We find this trade-off between what runs offline and what runs online. Sometimes the model size is huge, so it cannot be hosted. At the end of the day, we empower the data scientists to build models that fit the data. I always used to say, "Don't approximate the problem, approximate the solution."

Q: How do you address scaling your models?

Distributed computing is very important, which we leverage a lot. For something like GPUs, we have some services on-premise, some on cloud platforms. We try to containerize everything as much as possible, so that auto-scaling seamlessly comes into play.

Q: Are there any recent recommender system papers on specific techniques or methods that you find intriguing that you are considering integrating into the workflow? or have you recently integrated specific methods?

We are seeing a lot of traction in exploration/exploitation because it's a well-understood problem. Contextual bandits and modeling of rewards in a rich way is something in which we are seeing very healthy payoffs. The exploit framework helps on revenue, but it also helps in presentation bias. We have shown on paper that it can be very useful for the business, and also make our model better because it tries to un-bias datasets.

Traditionally, recommenders may show the top 10 candidates and users interact with only the top 10 candidates, and the bottom or 20 or 30 never get a chance – unless some items go out of stock. But if the models explore/exploit, we continuously do online learning where if something performs really well, but the variance is very high, we quickly show it more to make sure if it works well. The same goes if something is bad, but you're not sure – we show more of it to make sure that we can weed it out. We do this in a very principled way using Thompson Sampling.

One area that I think has a lot of promising research is in self-supervised learning. When you look at the site, we know someone viewed it ... maybe it took 10 minutes for them to add the cart and 30 minutes to check out, and we know that evolution. The store data is "This customer bought this item:" We really didn't know what made them buy it, we didn't know the order of items. Previously the team would need to correlate across omni-channel data, attempting to marry these two sources of data. One comes as a basket of items, the other comes with clear intent and how intent translates to an action. Then we can have a richer understanding of e-commerce data, but with storyline data as it changes. It's a great problem to marry these two sources and normalize them.

EVEN OLDRIDGE

Merlin Engineering Lead

NVIDIA

Q: What is your current role? What are you responsible for?

I lead the Merlin engineering effort. Merlin is NVIDIA's play towards recommender systems. It started out as a focus on deep learning-based recommender systems, which is a passion of mine. One of the reasons that I was brought on board was looking into how you make deep learning-based recommenders work well in the GPU. There was a perception in the community at the time that recommender systems didn't fit on GPU hardware. And really, it was true to a certain extent. The hardware back then, and when I say back then, I'm talking three, four years ago, didn't have a lot of memory. 11 gigs was the upper end, and recommender systems are very memory constrained. The embeddings that represent the users and items make up the vast majority of a recommender system model.

When you think about the deep learning frameworks and when they were built, it was during the era of computer vision. And so the acceleration of those types of workflows was the priority and still is in a lot of cases. But the frameworks are starting to come around to the idea that recommender systems are a really common use case. The two companies behind the major frameworks, TensorFlow and PyTorch respectively, their business models are based on recommendation, so they see it as being important. They're doing a lot of it internally, but the tools originally weren't built with recommendation in mind.

My team works on improving these tools so that they fit the recommendation use case. Some of this is within the frameworks, like the accelerated data loaders we've built, but more and more we're expanding to include all of the components that make up a recommender system.

Q: What was your and your team's journey from deep learning to "the broader play"?

I started as an individual contributor, trying to do what I can to bridge the gap between RAPIDS™, which is our framework for doing the Python data science ecosystem on GPU, and the deep learning frameworks. And so my first six months at NVIDIA started out building that bridge. In the process, I was able to create a data loader for PyTorch that sped up deep learning training by about 15X in PyTorch. That individual work led to growth. I got one engineer who transferred

in and so we were working together, and I was leading the project. And the two of us became four, and four became eight, and eight, we're now up to 10, 11, and growing the team and looking for new members. The NVTabular side was our focus on the ETL and on accelerating the training frameworks. And about a year in, Jensen partnered us up with HugeCTR, a recommender system frameworks team building a new framework in China. That was the birth of Merlin.

The work the HugeCTR team does is a great example of NVIDIA's focus on the upper echelon of recommender systems. The current project they're working on is a 100 terabyte model. That's bigger than any customer we know has, but the customers are going to get there soon. And when you get to that scale, you really have to be smart and think, and they're looking at what teams are doing and are publishing in terms of the open source space and trying to integrate that into the product and they're also providing that guidance to other teams. Some teams adopt the technology just by integrating it into their own stack. At that level, you can, and I think the work that's gone on, a lot of it's been at that upper end and focused on that, and it makes sense. There's a lot of business there. The scale of companies working on recommenders, it varies dramatically. And the compute required at the upper end, it dwarfs the compute required at the lower end.

But we're really starting to shift that and think end to end in that journey, in the sense that when I started, it was all about deep learning-based recommenders and now, we're really just trying to solve the challenge of how do teams build good recommender systems? How can we make that easier? How can we provide not necessarily the final solution, but a solution at each stage of the pipe, something the team refers to as our omakase, the chef's tasting, where you've got, "Here's the different components that we recommend. Here's the chef's menu. And you can make substitutions, and that's totally fine, but we think that you're going to need a starter and a main and a salad and a soup and a dessert, and these are the wines that we pair with that, and that's how we've laid it out. End to end solutions that demonstrate how you build a recommender system with our tools provides a solid foundation that will allow the teams that are trying to build recommenders, and most importantly, get them into production, to move forward and to be able to build.

Q: What is the main focus of the team?

The team has three main focuses:

The first is making recommender systems easier to build. We're trying to make tools that make it easier to do the feature engineering, tools that make it easier to do the modeling, tools that make it easier to do anything that involves getting that recommender model ready for production.

The second thing is how do you get those models into production? And that is a huge barrier for a lot of companies. It's not easy, especially on the deep learning side, but even a basic model for collaborative filtering, getting something into production and then properly monitoring it and making sure that it's not going off the rails and making sure that you're tracking all the things you should be tracking. It's immensely complex. And so the second thing that the team works on is making it easy to deploy into production.

And then the third aspect is making sure it's performant on the GPU. A lot of teams in NVIDIA, that's their number one priority. Whatever we're building is performed on the GPU. That's job number one, and I don't think it is for us. In my opinion, it's the building the tools, and in many cases, the deploying to production is probably the hardest part of the pipeline and the greatest opportunity. And so I think that's really where our focus is shifting. It's just so hard, and the easier we can make it the more successful our customers will be.

In terms of the deep learning versus less complex ML models, that's a journey that a lot of companies want to make. I've tried to do that myself a number of times. I've worked with other people. I've talked with people... It's consistently a challenge that the data scientist has pulled the data set, and they've got a model and it looks good offline, and they're excited to try it. And then there's just this mountain to climb to get that model deployed into production. And the bigger companies like the Facebooks of the world, they solve that, usually, with a big engineering team.

Q: Over and beyond the deep learning side of the modeling, are there other particular areas of focus? For instance, I know a lot of people have been talking about session-based RecSys.

Session-based RecSys is a very interesting space. It's a little bit newer in the RecSys space. I see a lot of companies in the Asia Pacific region, in particular, who are leveraging it, and they're showing massive gains and performance consistently at ACM RecSys the last couple of years. They come up with these great, session-based models. They're running them in production. They're showing big performance improvements in their RecSys papers.

This year our team is presenting a paper at ACM RecSys on our Transformers4Rec library, which we think is a nice contribution to the field, and the goal or the impetus behind that library was really starting from the idea that a lot of ideas in the RecSys space and session-based recommenders in particular come from the NLP domain. But there's this lag that happens between the NLP space and the RecSys space. And there's two orders of magnitude more research going on in NLP than there is in RecSys, despite the fact that the amount of compute is probably reverse that in terms of the real world production. And it's largely due to the lack of data sets.

Q: Based on what you've seen, leading this team, working with customers, looking at the problems in depth, what kind of recommendations would you have for someone who's just starting out? They're starting from zero to embrace building out recommender systems in their practice, and what's a good way to get a good foot on the ground?

I think that the key is to start simple and iterate. There's a tendency to want to build the latest, greatest deep learning model, I think, amongst data scientists universally. And starting out with something more straightforward and simple to make sure that the signals are there. Good data is always better than a good model, in my experience, and I think that's a lesson that data scientists often learn the hard way. But I feel like it's a pretty universal state and keeping your data maintained and clean, and then doing feature discovery often will lead to way more benefit than starting from I'm going to try this latest model perspective.

Really, the main thing is to figure out a process so that you can go from, "Here's the model, and here's how we're going to iterate on it and the output of that is going to be a model of this type." And that model is going to go into production in this sense, and thinking about and talking to the teams that are doing that production aspect first so that you understand what's being used and how that can be deployed, and what the constraints are there.

And we're trying to, with Merlin, provide that framework so that you can start with the simple models and then build up over time to more complex. It is a gap in the field. It's currently very hard to make the transition from traditional ML to deep learning, and a lot of companies struggle with that stage and have to spin up a completely different pipeline to do the deep learning-based work and it's complex. So thinking about that lifecycle and your MLOps and the iteration,

figuring out how to iterate quickly before you get to the point where you're building one or two or three models. If you can get to the point where you've got that system in place, that you've got the monitoring in place, that you understand your models really well, that you're tracking really well, that you understand the data signals, that the data's clean. Modeling is one of the last steps in the sense that you should be optimizing and it's often the first place people start.

So to answer your question more succinctly, we hope that Merlin is a good place to start, and we're certainly working hard to make it so. Developing and deploying recommender systems is an incredibly complex task today. ML in general is, but RecSys even more so. That's shifting though, and we're very focused on Merlin as an easy to use, easy to deploy, performant framework.

ABOUT THE AUTHOR

Paco Nathan

Managing Partner at Derwen, Inc

Known as a "player/coach", with core expertise in data science, cloud computing, natural language, graph technologies; ~40 years tech industry experience, ranging from Bell Labs to early-stage start-ups. Advisor for Amplify Partners, Recognai, KUNGFU.AI. Lead committer PyTextRank, kglab. Formerly: Director, Community Evangelism @ Databricks and Apache Spark. Cited in 2015 as one of the Top 30 People in Big Data and Analytics by Innovation Enterprise.

LEARN MORE

To take a deeper dive into NVIDIA Merlin, visit: developer.nvidia.com/nvidia-merlin

NVIDIA