Recommend a Movie, Win a Million Bucks

Can a computer learn to choose movies you're sure to like? Artificial intelligence took on this real-world problem in the Netflix contest. A Caltech alum recounts his adventures in the quest for the million-dollar grand prize.

Many a Caltech PhD goes into research, but for the better part of a year I did so alone and unemployed—and, yes, often in my pajamas. Yet, without ever leaving my apartment, I ended up on an 11-country team chasing the million-dollar Netflix Prize in what Caltech professor Yaser Abu-Mostafa [PhD '83] has aptly described as the Super Bowl of machine learning. There were 5,169 teams competing, and with one day to go we were in first place.

Modeled after the 2003 X Prize challenge to build a privately funded spaceship, the Netflix Prize was intended to harness the creativity of computer scientists and statisticians worldwide. Humans (most of us anyway), instinctively learn from experience to recognize patterns and make predictions, but designing machines that "learn" in an even remotely similar manner will provide fodder for PhD theses for decades to come. The Netflix challenge, to build a better method for predicting a customer's future movie preferences based on past movie ratings, ultimately led to many discoveries, including profound advances in statistical modeling—as well as the fact that fans of the TV series *Friends* tend to dislike Stanley Kubrick movies.

A visitor to the Netflix website looking for something to rent faces a sea of more than 100,000 choices. Cinematch, the website's recommendation system, helps you sort through them. Cinematch bases its suggestions on a staggeringly huge database of movie ratings collected whenever one of Netflix's 10-million-plus subscribers clicks on one of the five "star" buttons below the thumbnail picture of each movie's cover art. The average customer has rated some 200 movies, so Cinematch has plenty of preference data to work with, but still, as of 2005,

Contrary to popular belief, self-employed computer gurus do occasionally spend time in the sun. Sill enjoys a balmy spring day on Chicago's lakefront.

By Joseph Sill

Netflix was not entirely happy with it. Some of its suggestions seemed pretty random.

The company had been tweaking the system for years, and finally Netflix founder and CEO Reed Hastings tried to fix things himself. He neglected his wife and family over the 2005 Christmas vacation while manipulating spreadsheets and trying to make sense of mountains of customer data. He soon concluded that he was in over his head, and Netflix eventually decided to look outside the company for help. On October 2, 2006, Netflix formally offered one million dollars to whoever submitted the best algorithm, as long as it beat Cinematch's performance by at least 10 percent.

Entrants were allowed to download a data set of more than 100,000,000 ratings spanning seven years; 480,189 customers, represented anonymously by ID numbers; and 17,770 movies and TV series. Each data point consisted of four pieces of information: the customer ID, the movie or TV show's title, the rating, and the date the rating was made. This was the "training set"—the data you let your computer puzzle over to tease out the patterns.

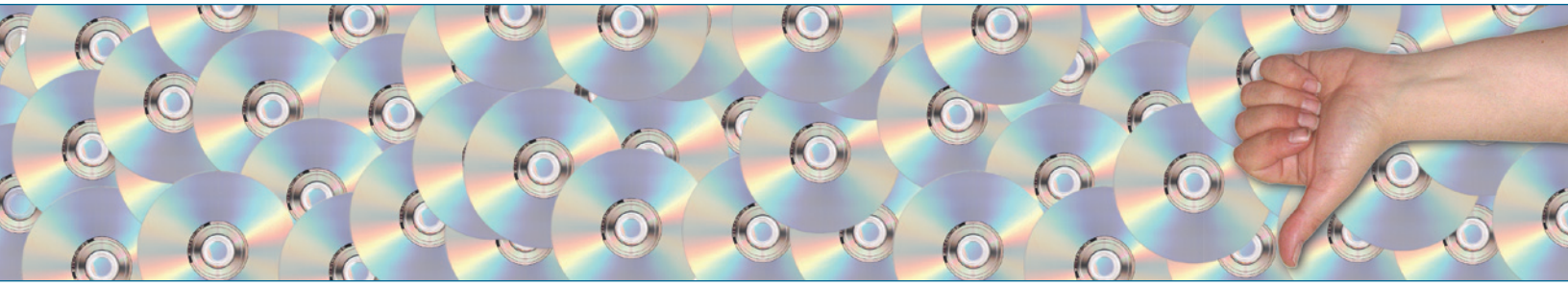
Some simple calculations showed that the highest-rated movies were the *Lord of the Rings* trilogy, with the final installment, *The Return of the King*, in first place with an average rating of 4.72 stars. Interestingly, every single one of the bottom 10 movies was a horror flick. In one case, this may have been intentional, since the sixth-lowest movie (1.40 stars) was titled *The Worst Horror Movie Ever Made*. Although this film came close to living up to—or down to—its name, dead last went to *Avia Vampire Hunter* at 1.29 stars. (A zero-star rating was not permitted.) But winning the contest would require much more complex mathematics.

The competition itself involved another 2.8 million data points where only the customer IDs, titles, and dates were revealed. This data had been randomly split into two subsets, the quiz set and the test set, and contestants were not told which data points belonged to what set. Teams were allowed to upload their predictions for the entire set to the Netflix Prize server once a day. The server then compared the predictions to the true ratings and calculated the root mean squared error, or RMSE, for each set. Loosely speaking, RMSE measures the average error. If an algorithm predicted three stars when the true rating was four, for example, the RMSE would be one. The teams' RMSEs on the quiz set were posted in rank order on a public web page known as the leaderboard. The RMSEs on the test set, which would determine who—if anyone—would win the million dollars, were known only to Netflix.

Things started out swiftly, with an improvement of more than 8.5 percent over Cinematch in the first 18 months. I had a highly demanding job as a quantitative analyst for a hedge fund at the time, so I was not following the action very closely. But my old mentor, Professor of Electrical Engineering and Computer Science Abu-Mostafa, was—he was using the Netflix Prize as the basis for a project-based machine-learning course. I visited him in early 2008 to help interview prospective grad students, and he filled me in. We'd been friends for more than 15 years, so his interest excited mine; but still, my job left me no free time.

My own career in machine learning had begun in Caltech's computation and neural systems program, which occupies the intersection of computer science and engineering with neurobiology. In it, neurobiologists use computers to help understand the brain, and engineers look to neurobiology for inspiration in designing machines. With a background in applied math, I was on the theoretical periphery, and only survived the two demanding lab courses through the patience and generosity of my unlucky lab partners. One final required us to determine the inner workings of an analog VLSI chip (a particularly esoteric device) by running various diagnostic tests. I passed only by divine intervention—the fabrication plant accidentally burned the chips, and the exam was cancelled. I'm much more comfortable working with data, which in the mid-'90s meant a few thousand data points if we were lucky.

The sheer enormity of the Netflix dataset was alluring, and in the summer of 2008 a career decision provided the time to dig into it. The full fury of that fall's market



Left: A CS 156b discussion section. Abu-Mostafa (seated, at left) and grad student Panna Felson (BS '09) look on as senior Constantine (Costis) Sideris holds forth. Grad student and teaching assistant Chess Stetson is sitting on the right.

Right: At the end of 2007, BellKor ruled the Netflix leaderboard.

THE NETFLIX PRIZE AS A TEACHING TOOL

The most important feature of the Netflix competition was not the size of the prize, one million dollars, (although that didn't hurt!) but rather the size of the data: 100 million points. Machine learning researchers are used to much smaller data sets, and Netflix provided several orders of magnitude more data than what we usually have to be content with. It was a machine learning bonanza.

Machine learning lives or dies by the data that the algorithm learns from. The whole idea of this technology is to infer the rules governing some underlying process—in this case, how people decide whether they like or dislike a movie—from a sample of data generated by that process. The data are our only window into the process, and if the data are not enough, nothing can be done but make guesses. That doesn't work very well.

I saw the Netflix data as an opportunity to get my CS 156 Learning Systems students to try out the algorithms that they had learned in class. With 100 million data points, the students could experiment with all kinds of ideas to their heart's content.

The first time I gave the Netflix problem as a class project, the competition was still going on. For the project, each team had to come up with its own algorithm and maximize its performance. Then all the algorithms would be blended to give a solution for the class as a whole. Like Joe, our hope was to rapidly climb the leaderboard with each submission.

Perhaps my biggest challenge as an instructor was to work out a way of ensuring that the various teams tried different techniques. As you will see, blending radically different solutions is key to getting good performance, so if everybody tried the same approach because it seemed to be the most promising technique known, blending the results of such duplicative efforts would not give a lot of improvement.

Therefore, I announced that a team's grade would not depend on their algorithm's individual performance, but on the incremental improvement in the class-wide solution when the algorithm was incorporated into the blend. This gave the students an incentive to explore the less-traveled roads that might offer a better chance of shining in the blended solution. There was great educational value in pushing people to venture "outside the box."

How did we do? Well, I registered two team names with Netflix. I told the students that if we did really, really well, we would submit under "Caltech." The other name was that of a rival school back east, and we would use it if we did really, really badly. It turned out that we fared neither too badly nor too well. Our effort gave about a 6 percent improvement over the original Cinematch system, so we did not officially submit it.

The second time the course was offered, the Netflix competition had already ended. There was no blending this time around, and the grading process was more of a judgment call.

With the benefit of hindsight, the class' performance was much better. One team was getting a weekly improvement that it took the teams in the actual competition months to achieve. In fact, at one point a guy in the class wanted to set an appointment to meet with me. He had the top individual score of 5.3 percent, so I jokingly told him that he needed to get to 6 percent before I would see him. Within a few days he had gotten to 6.1 percent.

—YA-M e&s

meltdown had not yet hit, and my employer was doing fine, but I was ready to move on from finance. I needed some time off, and I needed a project to keep my skills sharp while I mulled things over. The Netflix Prize was perfect.

GETTING UP TO SPEED

The contest had now been going on for almost two years, so I had a lot of catching up to do. But I had ample time, now that I had quit my job, and there was a road map to the top of the leaderboard. To keep things moving, Netflix was offering an annual $50,000 "Progress Prize" to the team with the lowest RMSE, as long as it represented at least a 1 percent improvement on the previous year's best score. To claim the prize, however, you had to publish a paper describing your techniques at a reproducible level of detail. The first Progress Prize had gone to BellKor, a team from AT&T Research made up of Yehuda Koren (now with Yahoo! Research), Bob Bell, and Chris Volinsky. All I had to do was follow their instructions, and in theory I should get similar results.

As I read BellKor's paper, I quickly realized that ironic quotation marks belonged around the phrase "all I had to do." Their solution was a blend of over 100 mathematical models, some fairly simple and others complex and subtle. Fortunately, the paper suggested that a smaller set of models—perhaps as few as a dozen—might suffice. Even so, nailing down the details would be no easy task.

Leaderboard

Display top 40 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
Grand Prize - RMSE <= 0.8563				
--	No Progress Prize candidates yet	--	--	--
Progress Prize - RMSE <= 0.8625				
1	BellKor	0.8700	8.56	2007-12-31 02:55:23
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				
2	KorBell	0.8712	8.43	2007-10-01 23:25:23
3	When Gravity and Dinosaurs Unite	0.8714	8.41	2007-12-24 12:59:34
4	Gravity	0.8725	8.29	2007-12-20 14:29:02
5	basho	0.8729	8.25	2007-11-24 14:27:00
6	Dinosaur Planet	0.8753	8.00	2007-10-04 04:56:45
7	Just a guy in a garage	0.8770	7.82	2007-12-21 18:19:12
8	M @UToronto A	0.8787	7.64	2007-09-30 20:41:54

Several other leading teams had also published papers, even though they were not compelled to do so, and some of their methods also looked promising. Three general approaches seemed to be the most successful: nearest neighbors, matrix factorization, and restricted Boltzmann machines.

Nearest-neighbor models are among the oldest, tried-and-true approaches in machine learning, so it wasn't surprising to see them pop up here. There are two basic variations—the user-based approach and the product-based approach.

Suppose the task is to predict how many stars customer 317459 would give *Titanic*. The user-based approach would look for like-minded people and see what they thought. This similarity is measured by calculating correlation coefficients, which track the tendency of two variables to rise and fall in tandem. However, this approach has some serious drawbacks. The correlations between every possible pairing of customers need to be determined, which for Netflix amounts to 125 billion calculations—a very heavy computational burden. Furthermore, e-commerce companies have found that people tend to trust recommendations more if the reasoning behind them can be explained. "Selling" a user-based nearest-neighbor recommendation is not so easy: "There's this woman in Idaho who usually agrees with you. She liked *Titanic*, so we think you will, too—trust us on this one."

For these reasons, product-based nearest-neighbor techniques are now much more common, and the Netflix contest was no exception. With 17,770 movies, as opposed to half a million customers, computing the correlations between all possible pairings in the training set was much more manageable, and the resulting table fit easily into a computer's RAM. These precomputed correlations led to odd insights—it's how I learned that fans of *Friends* tend to dislike Stanley Kubrick. (If you correlate Netflix's catalog with the ratings given to a DVD of a season of *Friends* and rank the results, *Dr. Strangelove* and *2001: A Space Odyssey* end up at the bottom.) Typically, only positive correlations are used—for instance, *Pulp Fiction* is highly correlated with other Quentin Tarantino movies like *Reservoir Dogs* and *Kill Bill,* as well as movies with a similar dark and twisted sensibility like *Fight Club* and *American Beauty*. Thus a product-based approach allows Netflix to say, "since you enjoyed *Fight Club*, we thought you might like *Pulp Fiction*."

Surprisingly, I found that negative product-based correlations were nearly as useful as positive ones. Although the connections were not as obvious, there was often a certain logic to them. For instance, *Pulp Fiction* and Jennifer Lopez romantic comedies like *The Wedding Planner* and *Maid in Manhattan* were strongly anticorrelated, and fans of the TV series *Home Improvement* generally disliked quirky movies such as *The Royal Tenenbaums*, *Eternal Sunshine of the Spotless Mind*, and *Being John Malkovich*. I ended up devising a negative-nearest-neighbors algorithm—a "furthest opposites" technique, if you will—that scored an RMSE of 0.9570, which was nearly as good as the 0.9513 of the original Cinematch software. I got perverse enjoyment out of imagining a customer being told that since he hated *Rambo III*, he might like *Annie Hall*.

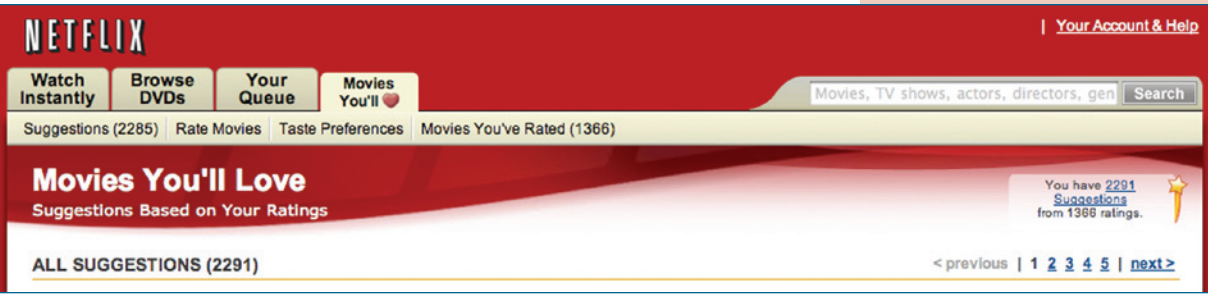
Nearest-neighbor systems improved greatly over the contest's first year, as teams delved into the math behind the models. These new twists represented significant progress in the design of recommendation systems, but a bigger breakthrough came in the form of another technique called matrix factorization.

Matrix factorization catapulted into prominence when "Simon Funk" suddenly appeared out of nowhere, vaulting into fourth place on the leaderboard. Funk, whose real name is Brandyn Webb, had graduated from UC San Diego at 18 with a computer science degree, and has since had a spectacular career designing algorithms. He was quite open about his method, describing it in detail on his website, and matrix factorization eventually became the leading technique within the Netflix Prize community.

Matrix factorization represents each customer and each movie as a vector, that is, a set of numbers called factors. The customer's predicted rating of that movie is the dot product of the two vectors—a simple mathematical operation involving multiplying each customer factor by each movie factor and summing up the result. Essentially, the movie vector encodes an assortment of traits, with

DOT PRODUCTS

Let's say that movie vectors have four factors: sex (S), violence (V), humor (H), and music (M). Maggie, a customer who has just watched *Brigadoon* and is looking for more of bonnie Scotland, has preferences S = 2, V = 1, H = 3, M = 5. If *Braveheart* scores S = 2, V = 5, H = 1, M = 1 on this scale, then the dot product predicting Maggie's likely reaction to Mel Gibson's gorefest is (2 × 2) + (1 × 5) + (3 × 1) + (5 × 1) = 17. Since the maximum possible score in this example is 100, she should probably give *Braveheart* a miss.



each factor's numerical value indicating how much of that trait the movie has. The customer vector represents the viewer's preferences for those traits. It's tempting to define the factors ahead of time—how much money the film grossed, how many Oscar winners are in the cast, how much nudity or violence it has—but in fact they are not predetermined in any way. They are generated by the model itself, inside the "black box," and only the model knows exactly what they mean.

"the temperature at which the brain begins to die" and intended as a Republican response) lowest. As the number of factors grew—beyond 100, in some cases!—they got much harder to interpret. The trends being discerned got more subtle, but no less real.

The third class of approaches, the restricted Boltzmann machine, is hard to describe concisely. Basically, it's a type of neural network, which is a mathematical

The default number of teams displayed on the web page was 40, so if I cracked the Top 40 I could at least point myself out to people easily.

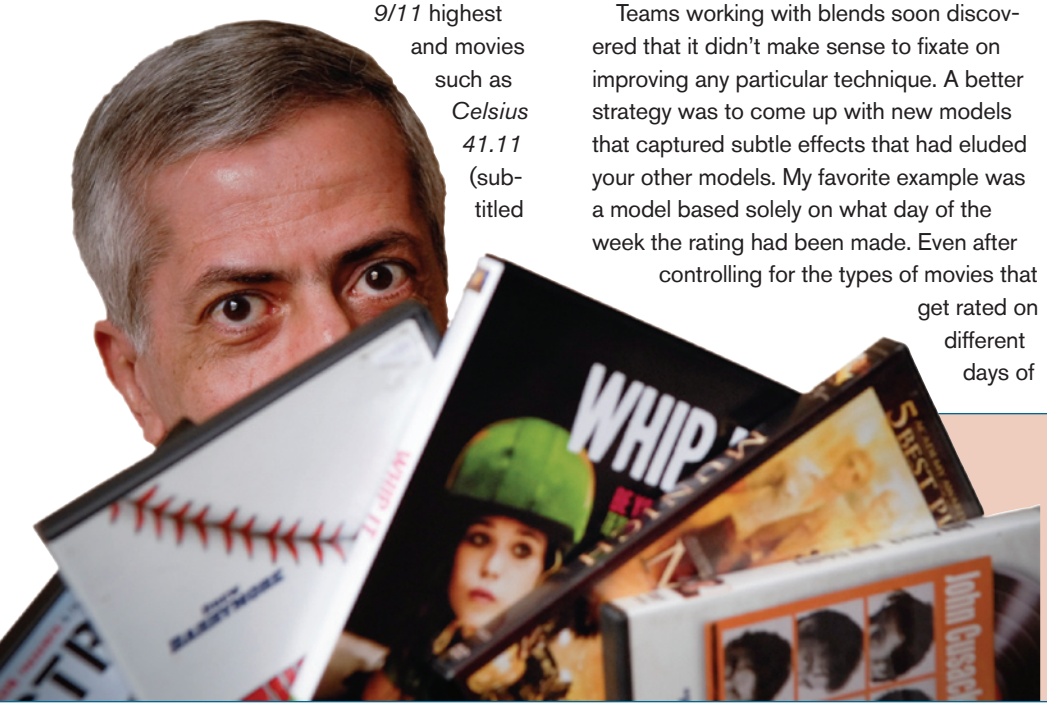
All we know is that as the model learns, it continually adjusts the factors' values until they reliably give the right output, or rating, for any set of inputs.

But if a model had just a handful of factors, sometimes their meanings leapt out. One such factor tagged teen movies: the *American Pie* series and *Dude, Where's My Car?* came out on top, and old-school classics like *Citizen Kane* and *The Bridge on the River Kwai* were at the bottom. Another loved romantic comedies such as *Sleepless in Seattle* and hated the *Star Trek* franchise. A third chose left-leaning films, placing Michael Moore movies such as *Fahrenheit 9/11* highest and movies such as *Celsius 41.11* (subtitled

construct loosely inspired by the interconnections of the neurons in the brain. As a computation and neural systems alumnus, I was glad to see neural networks well represented.

BellKor's $50,000 winning blend incorporated numerous twists and tweaks of all three major techniques as well as several less-prominent methods. This blending approach became standard practice in the Netflix Prize community. The process of blending predictions was itself a meta-problem in machine learning, in that the blending algorithm had to be trained how to combine the outputs of the component models.

Teams working with blends soon discovered that it didn't make sense to fixate on improving any particular technique. A better strategy was to come up with new models that captured subtle effects that had eluded your other models. My favorite example was a model based solely on what day of the week the rating had been made. Even after controlling for the types of movies that get rated on different days of the week, movies were more likely to get a bad rating on Mondays and more likely to get a good rating on weekends. On its own, this model performed horribly, but if none of the other models took the day of the week into account, this tiny but statistically significant signal could lead to a noticeable boost in the ensemble's accuracy.



Abu-Mostafa uses telepathy to pick movies for you. Machine learning systems can't do that yet.

BLENDED MODELS, BLENDED TEAMS

By the fall of 2008, having familiarized myself with all the major techniques and the art of blending large numbers of models, I set out to climb the leaderboard. I started with what I thought was a modest goal—the default number of teams displayed on the web page was 40, so if I could crack the Top 40 I could at least point myself out to people easily. Although my girlfriend was being very supportive, I wanted a tangible achievement to show for my solitary efforts hunched over the desk in the living room of my Chicago apartment, pounding away on my laptop. But with more than 5,000 teams competing, landing in the Top 40 meant being in the 99th percentile. I should have realized it wouldn't be so easy.

For one thing, there were subtleties unmentioned in the papers that made the difference between a working model and a failure. Deducing these tricks on my own sometimes took weeks at a time. Another challenge was weighing how much time to spend on known methods versus inventing original approaches. I probably spent too much time trying for a home run—a stupendous, brand-new technique.

Still, I scored one triumph. When blending their models, most of my competitors relied on linear regression, a time-honored technique whose many virtues include a straightforward, analytically exact procedure for obtaining the best fit to the data. My intuition suggested that the blend should be adaptive, depending on such side information as the number of ratings associated with each customer or movie. By making every model's coefficients a linear function

of these side variables, I got a significant boost in accuracy while still retaining linear regression's key merits.

Despite this breakthrough, come January 2009 I was still hovering just below where I wanted to be, stuck in the mid-40s. I would often peruse the top 10 with a mixture of respect and jealousy. One day, a new team suddenly appeared there—the Grand Prize Team, a name that struck me as presumptuous and cocky. When I read about them, though, I was intrigued. GPT's founders—a team of Hungarian researchers called Gravity, and a team of Princeton undergrads named Dinosaur Planet—had merged in 2007. That team, dubbed When Gravity and Dinosaurs Unite, had very nearly won the first Progress Prize, but was overtaken by BellKor in the final hours.

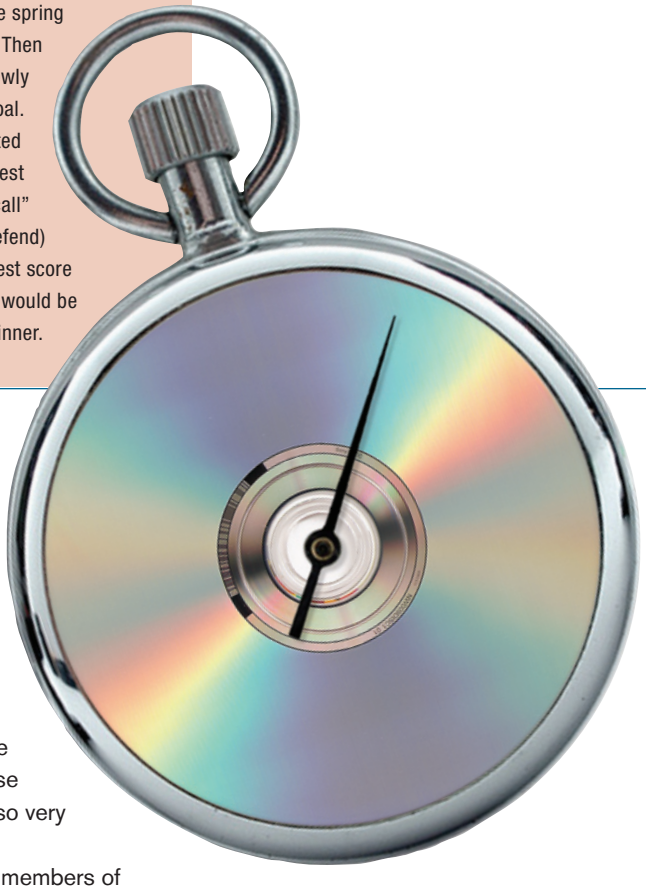
GPT issued a standing invitation to all comers to join them, but in order to be admitted the applicant had to demonstrably improve GPT's score—not an easy task. As of GPT's founding, their RMSE stood at 0.8655. The million-dollar goal was 0.8563, so only 0.0092—or 92 basis points, as they were called—remained to go.

Shaving off just one basis point was excruciatingly hard, and GPT's offer reflected this. The original members would only claim one-third of the prize, or $333,333, and the remainder would be split among the new members in proportion to their basis-point contribution. Thus one basis point, the smallest measurable improvement, was worth almost $7,000. This may seem overly generous, but it reflected how difficult progress had become. Weeks or even months would go by with nothing happening at the top of the leaderboard, and a boost of just a few basis points was cause for celebration. The old 80/20 rule—that 80 percent of the payoff comes in the first 20 percent of the work—was in full force.

My advancement as a lone wolf stymied, I crossed my fingers and sent my adaptive-blending code to GPT captain Gábor Takács. Reading the email I got in response was the most satisfying moment I had yet experienced—I'd boosted their score by more than 10 basis points! GPT's next official submission to Netflix a few days later jumped from 0.8626 to 0.8613 on the strength of my contribution, leapfrogging a few other entrants in the process. As of February 2009, I was suddenly a significant shareholder on a leading team.

However, first place belonged to a merger of BellKor and two Austrian graduate students calling themselves Big Chaos. The amalgam, BellKor in Big Chaos, had won the second Progress Prize in October 2008, with a score of 0.8616. They had since climbed to 0.8598, and there they had ground to a halt. It seemed like that figure was chiseled in stone—so close to the magic number, and yet so very far away.

Time was running out in the spring of 2009, but nobody knew it. Then suddenly, on June 26, a newly formed team reached the goal. The competition promptly shifted into overdrive, as the contest included a one-month "last call" for all comers to take (or defend) the lead. The team with the best score at the end of that month would be declared the winner.



My stake as one of a dozen members of GPT gave me newfound motivation, as did the contrast between our leap and our competitors' glacial progress. We were still underdogs, but underdogs with momentum on our side. Over the next few months, I found some other side variables I could exploit, and we crept up the leaderboard. A new recruit from Israel, Dan Nabutovsky, boosted our score by another six basis points.

Our collaborative style varied. Some GPT members (or prospective members) simply sent Gábor their predictions. He'd add them to the mix and see whether the overall blend improved—something he could do without even knowing how the new models worked. Gábor and I, however, were collaborating at a much deeper level, exchanging code and debating which approaches were most likely to give us a boost.

Meanwhile, another underdog was also making a run. Calling themselves Pragmatic Theory, Martin Chabbert and Martin Piotte of Quebec had been steadily rising up the leaderboard. Neither of them had any formal training in machine learning, and both were holding down full-time jobs. Nonetheless, by March 2009 they had surpassed BellKor in Big Chaos and taken the lead with a score of 0.8597. After a bit of jockeying, the leaderboard settled into a new equilibrium, with the two teams tied at 0.8596. As the weeks dragged on, GPT narrowed the gap, but the front-runners continued to inch ahead. I'd check the standings every day, and often several times a day, wincing on the rare days when a new score was posted. Their progress was so slow that I felt we had plenty of time. I was wrong.

ENDGAME

Late afternoon on Friday, June 26, I checked the leaderboard, and what I saw felt like a punch in the gut. A new team—BellKor's Pragmatic Chaos—had broken the million-dollar barrier, with a score of 0.8558. They hadn't won yet, though. The contest rules provided for a one-month "last call" during which everyone had a final shot at the prize, and the team with the best score at the end of this period would be declared the winner. So we weren't toast, but things didn't look good. We were 36 basis points behind, a gap that felt as wide as the Grand Canyon. We'd been running an ultramarathon—months-long for me; years-long for some of my teammates—and only the day before, we'd been on the heels of the leaders. Now, just short of the finish line, we looked up and found they were so far ahead that we had to squint to see them.

We decloaked on Saturday afternoon. With less than 24 hours to go, we were on top.

With so little time left, GPT became much more collaborative, with most members exchanging detailed information about their algorithms via email and chat sessions. Everyone else was scrambling to catch up as well. A few leaders were on a quixotic quest to win on their own, but many could see that there was strength in numbers. During the first two weeks of the final month, another coalition formed. Vandelay Industries, the name of a fake company from an episode of *Seinfeld*, and Opera Solutions, a real consulting firm, united and began aggressively recruiting anyone in the top 100. Soon their score was nearly the equal of ours, but even so, we were both still far behind. It was clear to all what we had to do. We negotiated a merger and a 50/50 split of the prize money, halving the value of my basis points. We were now a group of 30 people, and since the technique of blending models is known in the trade as ensemble learning, we named ourselves The Ensemble.

We worked hard but in stealth, keeping our new name off the leaderboard. BellKor's Pragmatic Chaos might have tried to recruit the uncommitted if they felt threatened, so we hoped to lull them into complacency by keeping our existence a secret. We had developed accurate ways to estimate our score in-house, and before long we determined that we could surpass the 0.8563 million-dollar barrier. Meanwhile, the opposition also inched ahead, to 0.8555. Little did they know that their lead was shrinking.

Our efforts intensified as the contest entered its final week. With team members in Europe, India, China, Australia, and the United States, we were working the problem 24/7. We now had thousands of models, and there were countless ways to blend them. We could even blend a collection of blends. In fact, our best solution was many-layered—a blend of blends of blends of blends. If we won, we'd have to document our methods before being awarded the prize, and I began to worry whether we'd be able to reconstruct what we had done. With so little time left, though, we decided to deal with that later.

The competition was scheduled to end on Sunday, July 26, 2009, at 1:42 p.m. Chicago time. That Thursday, we obtained a thrilling result: 0.8554. We could take the lead! Ah, but should we do so publicly? As we phrased it in our email discussions, should we decloak? Our members' agendas varied—some of us were all about winning, while others were more interested in the publicity. If we decloaked too soon, we might spur BellKor's Pragmatic Chaos to work extra hard or go on a recruiting binge. On the other hand, our window of opportunity might not last. If our next submission was merely the runner-up, we'd get less media attention than if we had suddenly seized first place. Ultimately, we decloaked on Saturday afternoon. With less than 24 hours to go, we were on top.

However, we were Number One with an asterisk. Remember that the leaderboard posted the quiz set's results, while the true winner would be the highest performer on the test set—a ranking known only to the Netflix engineers running the contest. Since both sets were drawn from the same data, the two scores should be close, but precisely how close? Our learning and tweaking process had been influenced in subtle ways by feedback from the quiz set—an example of what's known in the education biz as "teaching to the test." What would happen now that the test itself had changed?

Our first-place debut brought a rush of adrenaline, but the game was far from over. We planned to make one last submission a few minutes before the contest ended. We worked frantically through Saturday night and into Sunday morning. With 20 minutes to go, BellKor's Pragmatic Chaos matched our quiz score of 0.8554. A cacophony of emails and chat messages ensued, and as the final minutes ticked away, Peng Zhou announced from Shanghai that he thought he could achieve 0.8553. We sent in his blend, retook the lead, and the deadline passed. We had finished first, but had we won?

Shortly after the final buzzer sounded, Netflix announced on the Netflix Prize discussion board that two teams had qualified, and that their submissions were being evaluated. We still didn't know who'd won, but at least we knew we'd achieved a 10 percent (or better!) improvement on the test set. We were relieved that most of our accuracy on the quiz set had carried over.

In previous years, the Progress Prize winners had been notified via email shortly after the deadline but well before any public announcement. We had a gentleman's agreement with our rivals that if either team received such an email, we'd notify the other. An agonizing 90 minutes passed, and finally, around 3:00 p.m., we got the email we dreaded. BellKor's Pragmatic Chaos had won.

I went for a long jog to clear my head. I was a little depressed, but I wasn't feeling so bad. It was safe to say that more of their original work was in our solution than vice versa, since they had been required to publish two papers in order to win the Progress Prizes. I was also in awe of Pragmatic Theory, who had pulled off a stunning victory in a field where they were newcomers working in their spare time. I couldn't begrudge them the victory.

Netflix's official stance for the next two months was that the contest had not yet been decided, as the winning software was still being validated. This led to a fair amount of confusion. We still held first place on the leaderboard, and it was entirely reasonable for a casual observer to assume that we had won. Netflix asked us to say only that we were happy to have qualified, which led to some awkward situations.

Netflix finally announced the winner at a

press conference in New York on September 21. It was there that we learned that we had, in fact, tied, with both teams scoring an RMSE of 0.8567 on the test set. However, BellKor's Pragmatic Chaos had sent in their submission 20 minutes before we did. The pain of losing by such a minuscule margin was somewhat allayed by being quoted in the *New York Times*, and I bought several copies as souvenirs.

The press conference was my first opportunity to meet both my teammates and my competitors. I asked the Pragmatic Theory duo whether they were ready to quit their jobs and become machine-learning researchers. I was only half joking, but Martin Piotte just shrugged and said he might take up another hobby instead. After less than 18 months in the field, and having published perhaps the year's most important paper, I guess he felt it was time to move on.

At the awards ceremony that followed, some members of BellKor's Pragmatic Chaos gave a talk outlining the advances made in their final push. It turned out that Chabbert and Piotte had used the dates on which the ratings had been made in a very creative way. A movie could be rated in one of two contexts: either within a few days of watching it (Netflix asks customers to rate the movies they've just rented), or when browsing the website and rating previously seen movies. If a customer rated dozens or hundreds of movies on a single day, it was a safe bet that most of those films had been seen a long time ago. However, certain movies got better ratings immediately after viewing, while others fared better in retrospect. I had done a little work along these lines myself, and had squeezed out an additional 1-basis-point share in GPT as a result, but evidently the effect was more significant than I realized. Pragmatic Theory modeled it in detail and got a significant accuracy boost as a consequence.

Another interesting advance came from Big Chaos's Austrians. Most contestants optimized each individual model's accuracy on its own, and then blended it into the collection and crossed their fingers, hoping that the overall prediction improved. Big Chaos had found a way to train the individual models to optimize their contributions to the blend, rather than optimizing their own accuracy.

The power of blending, which the contest demonstrated over and over again, was the Netflix Prize's take-away lesson. You could even reverse-engineer a blend of models, using the results to design an "integrated" single model that incorporated the effects captured by the various simpler models. Indeed, Pragmatic Theory created a single model that scored 0.8713, equal to the RMSE of the 100-model blend that had won BellKor the first Progress Prize.

Although not all of the techniques involved in the winning solution have been incorporated into the working version of Cinematch, many of them have, and Netflix has seen increased customer loyalty since implementing these advances. Hastings, the Netflix CEO, said in the *New York Times* that the contest had been "a big winner" for the company. As for myself, I didn't quite finish as a winner in the formal sense, but I'm still thrilled with how the experience turned out. The contest has led to actual paid consulting work, and a group of my teammates and I are writing a paper on some of our techniques. Many, many other papers have and will come out of the contest, to the great benefit of the broader machine-learning research community. Only one team won the million dollars, but the Netflix competition ended up producing prizes of many kinds. E&S

Joseph Sill is an analytics consultant. He earned his Caltech PhD in 1998, and a BS in applied math from Yale in 1993, where he was elected to Phi Beta Kappa. Before competing for the Netflix Prize, he had worked for Citadel Investment Group and NASA's Ames Research Center.

This article was edited by Douglas L. Smith.

It ain't over till it's over, as Yogi Berra used to say. Well, it's finally over, and 20 minutes made all the difference.