Testimony before the Senate Subcommittee
on Communications, Technology, Innovation,
and the Internet Hearing on

Optimizing for Engagement: Understanding the Use of Persuasive Technology on Internet Platforms

25 June 2019

Statement of
Stephen Wolfram

Founder and Chief Executive Officer
Wolfram Research, Inc.

About Me

I have been a pioneer in the science and technology of computation for more than 40 years. I am the creator of Wolfram|Alpha which provides computational knowledge for Apple's Siri and Amazon's Alexa, and is widely used on the web, especially by students. I am also the creator of the Mathematica software system, which over the course of more than 30 years has been used in making countless inventions and discoveries across many fields. All major US universities now have site licenses for Mathematica, and it is also extensively used in US government R&D.

My early academic work was in theoretical physics. I received my PhD in physics at Caltech in 1979 when I was 20 years old. I received a MacArthur Fellowship in 1981. I was on the faculty at Caltech, then was at the Institute for Advanced Study in Princeton, then moved to the University of Illinois as Professor of Physics, Mathematics and Computer Science. I founded my first software company in 1981, and have been involved in the computer industry ever since.

In the late 1980s, I left academia to found Wolfram Research, and have now been its CEO for 32 years. During that time, I believe Wolfram Research has established itself as one of the world's most respected software companies. We have continually pursued an aggressive program of innovation and development, and have been responsible for many technical breakthroughs. The core of our efforts has been the long-term development of the Wolfram Language. In addition to making possible both Mathematica and Wolfram|Alpha, the Wolfram Language is the world's only full-scale computational language. Among its many implications are the ubiquitous delivery of computational intelligence, the broad enabling of "computational X" fields, and applications such as computational contracts.

In addition to my work in technology, I have made many contributions to basic science. I have been a pioneer in the study of the computational universe of possible programs. Following discoveries about cellular automata in the early 1980s, I became a founder of the field of complexity theory. My additional discoveries—with implications for the foundations of mathematics, physics, biology and other areas—led to my 2002 bestselling book *A New Kind of Science*. I am the discoverer of the simplest axiom system for logic, as well as the simplest universal Turing machine. My Principle of Computational Equivalence has been found to have wide implications not only in science but also for longstanding questions in philosophy.

My technological work has made many practical contributions to artificial intelligence, and the 2009 release of Wolfram|Alpha—with its ability to answer a broad range of questions posed in natural English—was heralded as a significant breakthrough in AI. My scientific work has been seen as important in understanding the theory and implications of AI, and issues such as AI ethics.

I have never been directly involved in automated content selection businesses of the kind discussed here. Wolfram|Alpha is based on built-in computational knowledge, not searching existing content on the web. Wolfram Research is a privately held company without outside investors. It employs approximately 800 people, mostly in R&D.

I have had a long commitment to education and to using the Wolfram Language to further computational thinking. In addition to writing a book about computational thinking for students, my other recent books include *Idea Makers* (historical biographies), and the forthcoming *Adventures of a Computational Explorer*.

For more information about me, see http://stephenwolfram.com

Summary

Automated content selection by internet businesses has become progressively more contentious—leading to calls to make it more transparent or constrained. I explain some of the complex intellectual and scientific problems involved, then offer two possible technical and market suggestions for paths forward. Both are based on giving users a choice about who to trust for the final content they see—in one case introducing what I call "final ranking providers", and in the other case what I call "constraint providers".

The Nature of the Problem

There are many kinds of businesses that operate on the internet, but some of the largest and most successful are what one can call *automated content selection businesses*. Facebook, Twitter, YouTube and Google are all examples. All of them deliver content that others have created, but a key part of their value is associated with their ability to (largely) automatically select what content they should serve to a given user at a given time—whether in news feeds, recommendations, web search results, or advertisements.

What criteria are used to determine content selection? Part of the story is certainly to provide good service to users. But the paying customers for these businesses are not the users, but advertisers, and necessarily a key objective of these businesses must be to maximize advertising income. Increasingly, there are concerns that this objective may have unacceptable consequences in terms of content selection for users. And in addition there are concerns that—through their content selection—the companies involved may be exerting unreasonable influence in other kinds of business (such as news delivery), or in areas such as politics.

Methods for content selection—using machine learning, artificial intelligence, etc.—have become increasingly sophisticated in recent years. A significant part of their effectiveness—and economic success—comes from their ability to use extensive data about users and their previous activities. But there has been increasing dissatisfaction and, in some cases, suspicion about just what is going on inside the content selection process.

This has led to a desire to make content selection more transparent, and perhaps to constrain aspects of how it works. As I will explain, these are not easy things to achieve in a useful way. And in fact, they run into deep intellectual and scientific issues, that are in some ways a foretaste of problems we will encounter ever more broadly as artificial intelligence becomes more central to the things we do. Satisfactory ultimate solutions will be difficult to develop, but I will suggest here two near-term practical approaches that I believe significantly address current concerns.

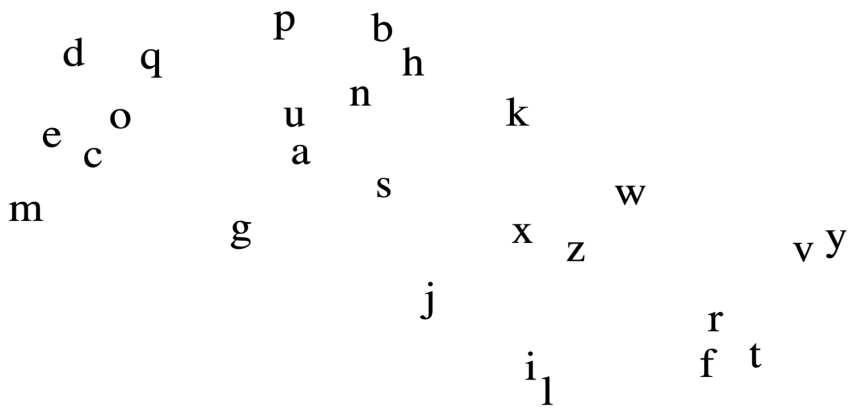
How Automated Content Selection Works

Whether one's dealing with videos, posts, webpages, news items or, for that matter, ads, the underlying problem of automated content selection (ACS) is basically always the same. There are many content items available (perhaps even billions of them), and somehow one has to quickly decide which ones are "best" to show to a given user at a given time. There's no fundamental principle to say what "best" means, but operationally it's usually in the end defined in terms of what maximizes user clicks, or revenue from clicks.

The major innovation that has made modern ACS systems possible is the idea of automatically extrapolating from large numbers of examples. The techniques have evolved, but the basic idea is to effectively deduce a model of the examples and then to use this model to make predictions, for example about what ranking of items will be best for a given user.

Because it will be relevant for the suggestions I'm going to make later, let me explain here a little more about how most current ACS systems work in practice. The starting point is normally to extract a collection of perhaps hundreds or thousands of features (or "signals") for each item. If a human were doing it, they might use features like: "How long is the video? Is it entertainment or education? Is it happy or sad?" But these days—with the volume of data that's involved—it's a machine doing it, and often it's also a machine figuring out what features to extract. Typically the machine will optimize for features that make its ultimate task easiest—whether or not (and it's almost always not) there's a human-understandable interpretation of what the features represent.

As an example, here are the letters of the alphabet automatically laid out by a machine in a "feature space" in which letters that "look similar" appear nearby:



How does the machine know what features to extract to determine whether things will "look similar"? A typical approach is to give it millions of images that have been tagged with what they are of ("elephant", "teacup", etc.). And then from seeing which images are tagged the same (even though in detail they look different), the machine is able—using the methods of modern machine learning—to identify features that could be used to determine how similar images of anything should be considered to be.

OK, so let's imagine that instead of letters of the alphabet laid out in a 2D feature space, we've got a million videos laid out in a 200-dimensional feature space. If we've got the features right, then videos that are somehow similar should be nearby in this feature space.

But given a particular person, what videos are they likely to want to watch? Well, we can do the same kind of thing with people as with videos: we can take the data we know about each person, and extract some set of features. "Similar people" would then be nearby in "people feature space", and so on.

But now there's a "final ranking" problem. Given features of videos, and features of people, which videos should be ranked "best" for which people? Often in practice, there's an initial coarse ranking. But then, as soon as we have a specific definition of "best"—or enough examples of what we mean by "best"—we can use machine learning to learn a program that will look at the features of videos and people, and will effectively see how to use them to optimize the final ranking.

The setup is a bit different in different cases, and there are many details, most of which are proprietary to particular companies. However, modern ACS systems—dealing as they do with immense amounts of data at very high speed—are a triumph of engineering, and an outstanding example of the power of artificial intelligence techniques.

Is It "Just an Algorithm"?

When one hears the term "algorithm" one tends to think of a procedure that will operate in a precise and logical way, always giving a correct answer, not influenced by human input. One also tends to think of something that consists of well-defined steps, that a human could, if needed, readily trace through.

But this is pretty far from how modern ACS systems work. They don't deal with the same kind of precise questions ("What video should I watch next?" just isn't something with a precise, well-defined answer). And the actual methods involved make fundamental use of machine learning, which doesn't have the kind of well-defined structure or explainable step-by-step character that's associated with what people traditionally think of as an "algorithm". There's another thing too: while traditional algorithms tend to be small and self-contained, machine learning inevitably requires large amounts of externally supplied data.

In the past, computer programs were almost exclusively written directly by humans (with some notable exceptions in my own scientific work). But the key idea of machine learning is instead to create programs automatically, by "learning the program" from large numbers of examples. The most common type of program on which to apply machine learning is a so-called neural network. Although originally inspired by the brain, neural networks are purely computational constructs that are typically defined by large arrays of numbers called weights.

Imagine you're trying to build a program that recognizes pictures of cats versus dogs. You start with lots of specific pictures that have been identified—normally by humans—as being either of cats or dogs. Then you "train" a neural network by showing it these pictures and gradually adjusting its weights to make it give the correct identification for these pictures. But then the crucial point is that the neural network generalizes. Feed it another picture of a cat, and even if it's never seen that picture before, it'll still (almost certainly) say it's a cat.

What will it do if you feed it a picture of a cat dressed as a dog? It's not clear what the answer is supposed to be. But the neural network will still confidently give some result—that's derived in some way from the training data it was given.

So in a case like this, how would one tell why the neural network did what it did? Well, it's difficult. All those weights inside the network were learned automatically; no human explicitly set them up. It's very much like the case of extracting features from images of letters above. One can use these features to tell which letters are similar, but there's no "human explanation" (like "count the number of loops in the letter") of what each of the features are.

Would it be possible to make an explainable cat vs. dog program? For 50 years most people thought that a problem like cat vs. dog just wasn't the kind of thing computers would be able to do. But modern machine learning made it possible—by learning the program rather than having humans explicitly write it. And there are fundamental reasons to expect that there can't in general be an explainable version—and that if one's going to do the level of automated content selection that people have become used to, then one cannot expect it to be broadly explainable.

Sometimes one hears it said that automated content selection is just "being done by an algorithm", with the implication that it's somehow fair and unbiased, and not subject to human manipulation. As I've explained, what's actually being used are machine learning methods that aren't like traditional precise algorithms.

And a crucial point about machine learning methods is that by their nature they're based on learning from examples. And inevitably the results they give depend on what examples were used.

And this is where things get tricky. Imagine we're training the cat vs. dog program. But let's say that, for whatever reason, among our examples there are spotted dogs but no spotted cats. What will the program do if it's shown a spotted cat? It might successfully recognize the shape of the cat, but quite likely it will conclude—based on the spots—that it must be seeing a dog.

So is there any way to guarantee that there are no problems like this, that were introduced either knowingly or unknowingly? Ultimately the answer is no—because one can't know everything about the world. Is the lack of spotted cats in the training set an error, or are there simply no spotted cats in the world?

One can do one's best to find correct and complete training data. But one will never be able to prove that one has succeeded.

But let's say that we want to ensure some property of our results. In almost all cases, that'll be perfectly possible—either by modifying the training set, or the neural network. For example, if we want to make sure that spotted cats aren't left out, we can just insist, say, that our training set has an equal number of spotted and unspotted cats. That might not be a correct representation of what's actually true in the world, but we can still choose to train our neural network on that basis.

As a different example, let's say we're selecting pictures of pets. How many cats should be there, versus dogs? Should we base it on the number of cat vs. dog images on the web? Or how often people search for cats vs. dogs? Or how many cats and dogs are registered in America? There's no ultimate "right answer". But if we want to, we can give a constraint that says what should happen.

This isn't really an "algorithm" in the traditional sense either—not least because it's not about abstract things; it's about real things in the world, like cats and dogs. But an important development (that I happen

to have been personally much involved in for 30+ years) is the construction of a computational language that lets one talk about things in the world in a precise way that can immediately be run on a computer.

In the past, things like legal contracts had to be written in English (or "legalese"). Somewhat inspired by blockchain smart contracts, we are now getting to the point where we can write automatically executable computational contracts not in human language but in computational language. And if we want to define constraints on the training sets or results of automated content selection, this is how we can do it.

Issues from Basic Science

Why is it difficult to find solutions to problems associated with automated content selection? In addition to all the business, societal and political issues, there are also some deep issues of basic science involved. Here's a list of some of those issues. The precursors of these issues date back nearly a century, though it's only quite recently (in part through my own work) that they've become clarified. And although they're not enunciated (or named) as I have here, I don't believe any of them are at this point controversial—though to come to terms with them requires a significant shift in intuition from what exists without modern computational thinking.

Data Deducibility

Even if you don't explicitly know something (say about someone), it can almost always be statistically deduced if there's enough other related data available

What is a particular person's gender identity, ethnicity, political persuasion, etc.? Even if one's not allowed to explicitly ask these questions, it's basically inevitable that with enough other data about the person, one will be able to deduce what the best answers must be.

Everyone is different in detail. But the point is that there are enough commonalities and correlations between people that it's basically inevitable that with enough data, one can figure out almost any attribute of a person.

The basic mathematical methods for doing this were already known from classical statistics. But what's made this now a reality is the availability of vastly more data about people in digital form—as well as the ability of modern machine learning to readily work not just with numerical data, but also with things like textual and image data.

What is the consequence of ubiquitous data deducibility? It means that it's not useful to block particular pieces of data—say in an attempt to avoid bias—because it'll essentially always be possible to deduce what that blocked data was. And it's not just that this can be done intentionally; inside a machine learning system, it'll often just happen automatically and invisibly.

Computational Irreducibility

Even given every detail of a program, it can be arbitrarily hard to predict what it will or won't do

One might think that if one had the complete code for a program, one would readily be able to deduce everything about what the program would do. But it's a fundamental fact that in general one can't do this. Given a particular input, one can always just run the program and see what it does. But even if the program is simple, its behavior may be very complicated, and computational irreducibility implies that there won't be a way to "jump ahead" and immediately find out what the program will do, without explicitly running it.

One consequence of this is that if one wants to know, for example, whether with any input a program can do such-and-such, then there may be no finite way to determine this—because one might have to check an infinite number of possible inputs. As a practical matter, this is why bugs in programs can be so hard to detect. But as a matter of principle, it means that it can ultimately be impossible to completely verify that a program is "correct", or has some specific property.

Software engineering has in the past often tried to constrain the programs it deals with so as to minimize such effects. But with methods like machine learning, this is basically impossible to do. And the result is that even if it had a complete automated content selection program, one wouldn't in general be able to verify that, for example, it could never show some particular bad behavior.

Non-explainability

For a well-optimized computation, there's not likely to be a human-understandable narrative about how it works inside

Should we expect to understand how our technological systems work inside? When things like donkeys were routinely part of such systems, people didn't expect to. But once the systems began to be "completely engineered" with cogs and levers and so on, there developed an assumption that at least in principle one could explain what was going on inside. The same was true with at least simpler software systems. But with things like machine learning systems, it absolutely isn't.

Yes, one can in principle trace what happens to every bit of data in the program. But can one create a human-understandable narrative about it? It's a bit like imagining we could trace the firing of every neuron in a person's brain. We might be able to predict what a person would do in a particular case, but it's a different thing to get a high-level "psychological narrative" about why they did it.

Inside a machine learning system—say the cats vs. dogs program—one can think of it as extracting all sorts of features, and making all sorts of distinctions. And occasionally one of these features or distinctions might be something we have a word for ("pointedness", say). But most of the time they'll be things the machine learning system discovered, and they won't have any connection to concepts we're familiar with.

And in fact—as a consequence of computational irreducibility—it's basically inevitable that with things like the finiteness of human language and human knowledge, in any well-optimized computation we're not going to be able to give a high-level narrative to explain what it's doing. And the result of this is that it's impossible to expect any useful form of general "explainability" for automated content selection systems.

Ethical Incompleteness

There's no finite set of principles that can completely define any reasonable, practical system of ethics

Let's say one's trying to teach ethics to a computer, or an artificial intelligence. Is there some simple set of principles—like Asimov's Laws of Robotics—that will capture a viable complete system of ethics? Looking at the complexity of human systems of laws one might suspect that the answer is no. And in fact this is presumably a fundamental result—essentially another consequence of computational irreducibility.

Imagine that we're trying to define constraints (or "laws") for an artificial intelligence, in order to ensure that the AI behaves in some particular "globally ethical" way. We set up a few constraints, and we find that many things the AI does follow our ethics. But computational irreducibility essentially guarantees that eventually there'll always be something unexpected that's possible. And the only way to deal with that is to add a "patch"—essentially to introduce another constraint for that new case. And the issue is that this will never end: there'll be no way to give a finite set of constraints that will achieve our global objectives. (There's a somewhat technical analogy of this in mathematics, in which Gödel's theorem shows that no finite set of axiomatic constraints can give one only ordinary integers and nothing else.)

So for our purposes here, the main consequence of this is that we can't expect to have some finite set of computational principles (or, for that matter, laws) that will constrain automated content selection systems to always behave according to some reasonable, global system of ethics—because they'll always be generating unexpected new cases that we have to define a new principle to handle.

The Path Forward

I've described some of the complexities of handling issues with automated content selection systems. But what in practice can be done?

One obvious idea would be just to somehow "look inside" the systems, auditing their internal operation and examining their construction. But for both fundamental and practical reasons, I don't think this can usefully be done. As I've discussed, to achieve the kind of functionality that users have become accustomed to, modern automated content selection systems make use of methods such as machine learning that are not amenable to human-level explainability or systematic predictability.

What about checking whether a system is, for example, biased in some way? Again, this is a fundamentally difficult thing to determine. Given a particular definition of bias, one could look at the internal training data used for the system—but this won't usually give more information than just studying how the system behaves.

What about seeing if the system has somehow intentionally been made to do this or that? It's conceivable that the source code could have explicit "if" statements that would reveal intention. But the bulk of the system will tend to consist of trained neural networks and so on—and as in most other complex systems, it'll typically be impossible to tell what features might have been inserted "on purpose" and what are just accidental or emergent properties.

So if it's not going to work to "look inside" the system, what about restricting how the system can be set up? For example, one approach that's been suggested is to limit the inputs that the system can have, in an extreme case preventing it from getting any personal information about the user and their history. The problem with this is that it negates what's been achieved over the course of many years in content selection systems—both in terms of user experience and economic success. And for example, knowing nothing about a user, if one has to recommend a video, one's just going to have to suggest whatever video is generically most popular—which is very unlikely to be what most users want most of the time.

As a variant of the idea of blocking all personal information, one can imagine blocking just some information—or, say, allowing a third party to broker what information is provided. But if one wants to get the advantages of modern content selection methods, one's going to have to leave a significant amount of information—and then there's no point in blocking anything, because it'll almost certainly be reproducible through the phenomenon of data deducibility.

Here's another approach: what about just defining rules (in the form of computational contracts) that specify constraints on the results content selection systems can produce? One day, we're going to have to have such computational contracts to define what we want AIs in general to do. And because of ethical incompleteness—like with human laws—we're going to have to have an expanding collection of such contracts.

But even though (particularly through my own efforts) we're beginning to have the kind of computational language necessary to specify a broad range of computational contracts, we realistically have to get much more experience with computational contracts in standard business and other situations before it makes sense to try setting them up for something as complex as global constraints on content selection systems.

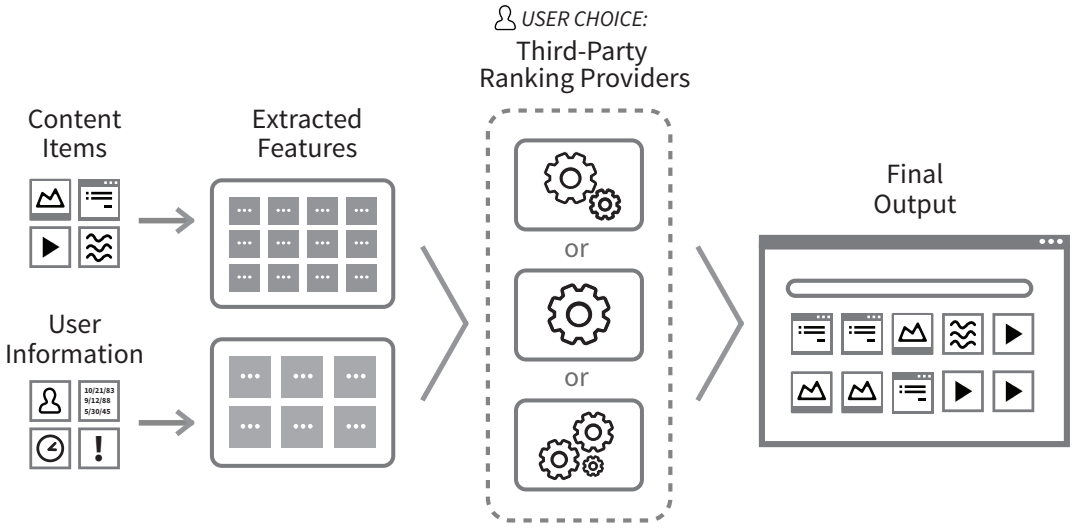
So, what can we do? I've not been able to see a viable, purely technical solution. But I have formulated two possible suggestions based on mixing technical ideas with what amount to market mechanisms.

The basic principle of both suggestions is to give users a choice about who to trust, and to let the final results they see not necessarily be completely determined by the underlying ACS business.

There's been debate about whether ACS businesses are operating as "platforms" that more or less blindly deliver content, or whether they're operating as "publishers" who take responsibility for content they deliver. Part of this debate can be seen as being about what responsibility should be taken for an AI. But my suggestions sidestep this issue, and in different ways tease apart the "platform" and "publisher" roles.

It's worth saying that the whole content platform infrastructure that's been built by the large ACS businesses is an impressive and very valuable piece of engineering—managing huge amounts of content, efficiently delivering ads against it, and so on. What's really at issue is whether the fine details of the ACS systems need to be handled by the same businesses, or whether they can be opened up. (This is relevant only for ACS businesses whose network effects have allowed them to serve a large fraction of a population. Small ACS businesses don't have the same kind of lock-in.)

Suggestion A: Allow Users to Choose among Final Ranking Providers



As I discussed earlier, the rough (and oversimplified) outline of how a typical ACS system works is that first features are extracted for each content item and each user. Then, based on these features, there's a final ranking done that determines what will actually be shown to the user, in what order, etc.

What I'm suggesting is that this final ranking doesn't have to be done by the same entity that sets up the infrastructure and extracts the features. Instead, there could be a single content platform but a variety of "final ranking providers", who take the features, and then use their own programs to actually deliver a final ranking.

Different final ranking providers might use different methods, and emphasize different kinds of content. But the point is to let users be free to choose among different providers. Some users might prefer (or trust more) some particular provider—that might or might not be associated with some existing brand. Other users might prefer another provider, or choose to see results from multiple providers.

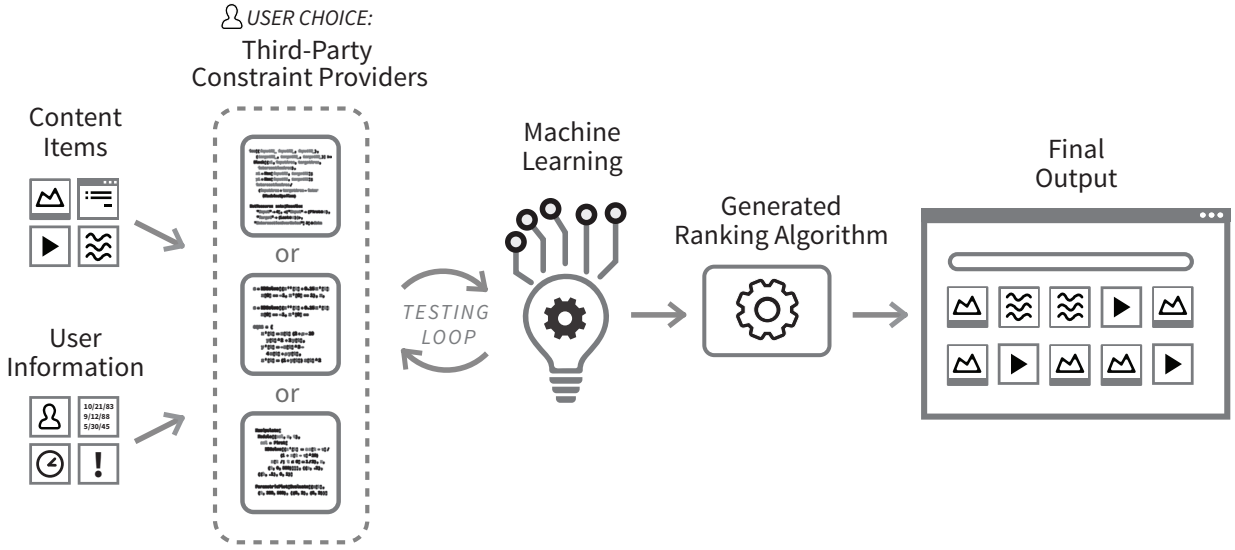
How technically would all this be implemented? The underlying content platform (presumably associated with an existing ACS business) would take on the large-scale information-handling task of deriving extracted features. The content platform would provide sufficient examples of underlying content (and user information) and its extracted features to allow the final ranking provider's systems to "learn the meaning" of the features.

When the system is running, the content platform would in real time deliver extracted features to the final ranking provider, which would then feed this into whatever system they have developed (which could use whatever automated or human selection methods they choose). This system would generate a ranking of content items, which would then be fed back to the content platform for final display to the user.

To avoid revealing private user information to lots of different providers, the final ranking provider's system should probably run on the content platform's infrastructure. The content platform would be responsible for the overall user experience, presumably providing some kind of selector to pick among final ranking providers. The content platform would also be responsible for delivering ads against the selected content.

Presumably the content platform would give a commission to the final ranking provider. If properly set up, competition among final ranking providers could actually increase total revenue to the whole ACS business, by achieving automated content selection that serves users and advertisers better.

Suggestion B: Allow Users to Choose among Constraint Providers



One feature of Suggestion A is that it breaks up ACS businesses into a content platform component, and a final ranking component. (One could still imagine, however, that a quasi-independent part of an ACS business could be one of the competing final ranking providers.) An alternative suggestion is to keep ACS businesses intact, but to put constraints on the results that they generate, for example forcing certain kinds of balance, etc.

Much like final ranking providers, there would be constraint providers who define sets of constraints. For example, a constraint provider could require that there be on average an equal number of items delivered to a user that are classified (say, by a particular machine learning system) as politically left-leaning or politically right-leaning.

Constraint providers would effectively define computational contracts about properties they want results delivered to users to have. Different constraint providers would define different computational contracts. Some might want balance; others might want to promote particular types of content, and so on. But the idea is that users could decide what constraint provider they wish to use.

How would constraint providers interact with ACS businesses? It's more complicated than for final ranking providers in Suggestion A, because effectively the constraints from constraint providers have to be woven deeply into the basic operation of the ACS system.

One possible approach is to use the machine learning character of ACS systems, and to insert the constraints as part of the "learning objectives" (or, technically, "loss functions") for the system. Of course, there could be constraints that just can't be successfully learned (for example, they might call for types of content that simply don't exist). But there will be a wide range of acceptable constraints, and in effect, for each one, a different ACS system would be built.

All these ACS systems would then be operated by the underlying ACS business, with users selecting which constraint provider—and therefore which overall ACS system—they want to use.

As with Suggestion A, the underlying ACS business would be responsible for delivering advertising, and would pay a commission to the constraint provider.

Although their detailed mechanisms are different, both Suggestions A and B attempt to leverage the exceptional engineering and commercial achievements of the ACS businesses, while diffusing current trust issues about content selection, providing greater freedom for users, and inserting new opportunities for market growth.

The suggestions also help with some other issues. One example is the banning of content providers. At present, with ACS businesses feeling responsible for content on their platforms, there is considerable pressure, not least from within the ACS businesses themselves, to ban content providers that they feel are providing inappropriate content. The suggestions diffuse the responsibility for content, potentially allowing the underlying ACS businesses not to ban anything but explicitly illegal content.

It would then be up to the final ranking providers, or the constraint providers, to choose whether or not to deliver or allow content of a particular character, or from a particular content provider. In any given case, some might deliver or allow it, and some might not, removing the difficult all-or-none nature of the banning that's currently done by ACS businesses.

One feature of my suggestions is that they allow fragmentation of users into groups with different preferences. At present, all users of a particular ACS business have content that is basically selected in the same way. With my suggestions, users of different persuasions could potentially receive completely different content, selected in different ways.

While fragmentation like this appears to be an almost universal tendency in human society, some might argue that having people routinely be exposed to other people's points of view is important for the cohesiveness of society. And technically some version of this would not be difficult to achieve. For example, one could take the final ranking or constraint providers, and effectively generate a feature space plot of what they do.

Some would be clustered close together, because they lead to similar results. Others would be far apart in feature space—in effect representing very different points of view. Then if someone wanted to, say, see their typical content 80% of the time, but see different points of view 20% of the time, the system could combine different providers from different parts of feature space with a certain probability.

Of course, in all these matters, the full technical story is much more complex. But I am confident that if they are considered desirable, either of the suggestions I have made can be implemented in practice. (Suggestion A is likely to be somewhat easier to implement than Suggestion B.) The result, I believe, will be richer, more trusted, and even more widely used automated content selection. In effect both my suggestions mix the capabilities of humans and AIs—to help get the best of both of them—and to navigate through the complex practical and fundamental problems with the use of automated content selection.