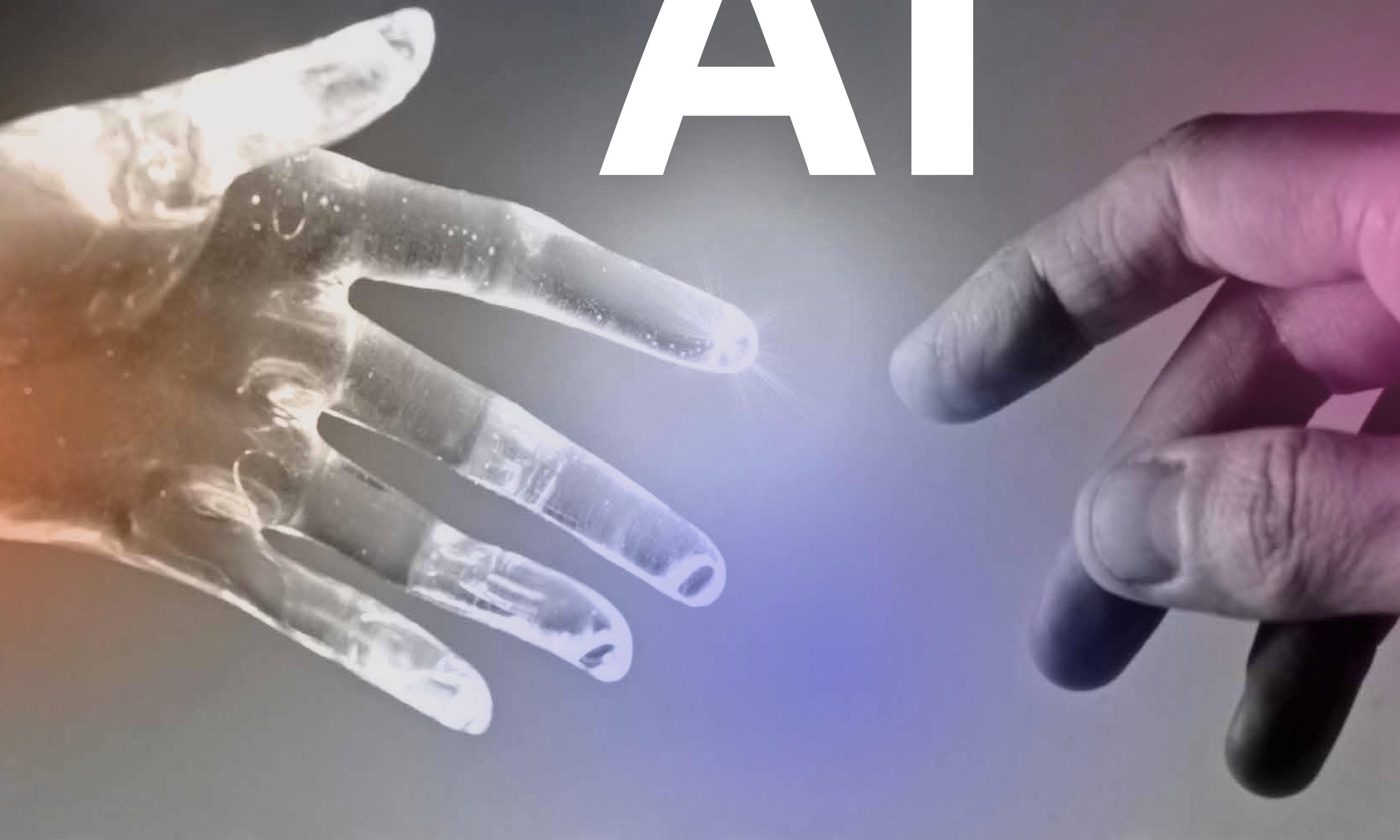
ML MAGAZINE

mlconference.ai

1/2023

The Latest in AI



Exploring the Future of ML

© Zsolt Biczo/Shutterstock.com

CONTENT

Alexander Goschin
Editor, ML Magazine

Gold Rush

Will an artificial intelligence soon dig for gold, so that we can comfortably put our feet up, relax and let the sun shine on our bellies while it does the tedious part of the work for us? Probably, eventually. But the comparison of current developments in AI and the American Gold Rush of the 19th century is obvious not because of that future ability, but because a similar mood prevails. Not only do we find ourselves, similar to the time of the American Gold Rush, in a certain social change and a global "Zeitenwende", but also because the rush tends to cloud our reason. The American Gold Rush of the 19th century serves as a great cautionary tale about the dangers of getting caught up in the ecstasy of gold and losing sight of what's possible and feasible.

Anyone who wants to deal with the current opportunities, possibilities and dangers of artificial intelligence must be alert, if he or she does not want to be misled by con men. By no means is artificial intelligence just a trendy topic or an empty marketing shell. Especially due to the progress of Large Language Models and generative AI in general, the topic deserves its new momentum and everybody who sees himself in areas around this topic is advised to deal with the new possibilities and dangers. But there is still the need to be levelheaded and to evaluate the current developments.

That is why we see it as our task to overcome some of this great clutter and confusion through our contributions. Because you, dear readers, should also deal with AI. At the same time, you should not fall into a rush to chase new sources of income but deal prudently with the new opportunities and perspectives. AI is here to stay and we, as users, societies and governments just started our journey on a long road to finding out, discussing and ultimately deciding, what role it should play in our collective and personal lives.

@mlconference

On Neurons and Data

Artificial General Intelligence: The most exciting project on the planet

Future AI Founder and CEO Charles Simon, BSEE, MSCs recently delivered the keynote address at the 15th Annual AGI Conference, AGI-22. The article below contains excerpts from his address, which covered the fundamentals of programming and neurons, what is known about how the human brain works, and how artificial general intelligence (AGI) will gradually emerge.

by Charles Simon

Simon is a nationally recognized entrepreneur and software developer, the author of “Will Computers Revolt? Preparing for the Future of Artificial Intelligence,” and the developer of Brain Simulator II, an AGI research software platform. For more information, visit Future AI’s homepage [1].

How Big is the Problem of Creating AGI?

AGI is the most exciting project on the planet. We get to do cool stuff with computers while probing the mysteries of the mind. AGI has been the holy grail of computer science since the 1950s. So today, I want to discuss what it would take to create AGI.

I’m going to approach AGI by scoping the size of the project. Some predict it’s going to be a hundred years before AGI occurs, others that it’s going to be next year. I fall on the earlier side of that spectrum, and want to explain why that is and propose some sort of structure for AGI. Then I’ll talk about the conclusions that can be reached from this structure, because it’s not the structure of today’s AI.

To broaden that concept and think about what overall general intelligence is – and try to get ourselves out of the pickle of creating many narrow projects, hoping they grow together – here’s another way to look at it. How big is the problem of creating AGI? Let’s consider the amount of information needed to define the human brain, our baseline for what general intelligence is.

When human DNA is added to the right kind of human stem cell, a person emerges. And this person not only has a human body and a human structure, but the start of human intelligence. And the total program size of the human genome? It is only 750 megabytes.

How much of that 750 megabytes of data is actually controlling the structure of the brain? In truth, we don’t know because the structure of any human part is largely dependent on its chemistry and we have a much better idea of how DNA goes to chemistry than we have about how it goes to structure. But somehow, the DNA defines what the structure of a person is and part of that defines the structure and chemistry of the brain. Once the brain is “turned on,” the brain’s structure immediately begins to modify itself to handle the data and experiences it receives. We can presume, though, that generalized intelligence is a direct outgrowth of the structure which is defined by our DNA.

The structure of the neocortex is what we think with. Maybe that represents only one percent of that DNA, maybe 10. We don’t know, but that means we might be able to define a complete AGI in a program that is as small as 7.5 megabytes. And 7.5 megabytes is a program that’s well within the scope of a team to write in a few years. Thus, the AGI problem is really one of what we are going to write, not that the underlying structure requires gigabytes to define. We just don't know what to write as the fundamental AGI algorithm.

Suppose you knew exactly what to write to make a human brain. Imagine you spend two years coding it then you start training it. If it's exactly like a human, after three years of training, it's going to have the capabilities of a three-year-old. Now, I like three-year-olds, but they are not marketable so we're going to need maybe 20 more years of training to get something that's really useful and generally intelligent. And we don't want to wait 25 years to find out if our idea was correct. Programming shortcuts are needed.

Knowing that, my approach to AGI is to go back to the only actual general intelligence we know about: the brain. I wrote the Brain Simulator so I could simulate neurons and learn what sorts of things you could do with them.

The brain is largely split into three areas. The brain stem is full of autonomic functions. We think it is hard-coded because nobody seems to have to take the time to learn how to get their heart to beat or how to breathe. The cerebellum is responsible for muscular coordination so I can talk and walk at the same time. Fifty-six billion neurons (about two-thirds of the brain in terms of neuron count) are involved in coordinating muscles and learning sequences of actions.

The neocortex is the thinking part of your brain. It's only 16 billion neurons and learns by a number of different mechanisms. All of the sensory stuff comes into the back of your brain – visual cortex, auditory cortex, etc. The forward part of your brain is involved in planning what to do and doing it. For example, the part of your brain that recognizes speech is in the sensory zone while the part that generates speech is in the action zone. Running across the top of your brain, roughly from ear to ear, are the motor cortex that controls the voluntary actions and the sensory cortex that receives the touch and pain signals from your body.

AGI need not mirror the brain's structure, but analyzing the brain provides a good inventory of the types of functionality a system will need to appear intelligent.

Looking Deeper at Neurons

Looking deeper at neurons, the brain has billions of them hooked together in a big network. The individual neuron generates a spike, which travels down the axon, goes to the synapses, connects to other neurons that it's connected to, and transmits a signal. The signal travels down the axon at approximately one meter per second. This is not an electrical signal, but a kind of a chain reaction of ions going from place to place. The neuron itself is extremely slow.

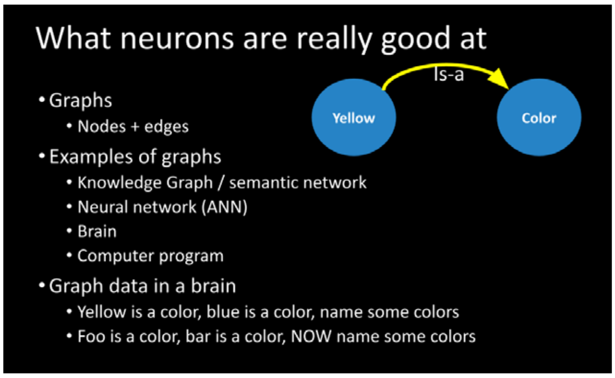
If you play with neurons in a simulator for a few years, you learn what neurons are good at doing and what they're not so good at doing. They're really good at signal differentiation (for example, in your eye you can detect boundaries of various colors or brightnesses with considerable accuracy). You can detect the arrival time of multiple signals and use this for directional sound.

You can have short-term memory in neuron firing or in the charge state in a neuron. This is really short-term – a memory you can establish with a single spike. You don't have to do any synapse adjustment. You can store something for a short period of time and the storage can happen very quickly. This storage mechanism might be used in detecting if objects are in motion. Your neurons need to store the current visual field very briefly in order to detect if anything has moved.

Synapses can also store memory in their configuration, the physical layout of the ions, and the synapse size. Once established, information in synapses requires no energy to keep indefinitely, so your brain can learn things now, set them in the configuration of the weights of your synapses, and store them indefinitely without burning energy. On the downside, it takes numerous neural spikes to adjust a synapse to a particular value so this method of storage is much slower.

Neurons are really slow – time frames of four milliseconds between spikes – which has a dramatic impact on the kinds of functions they can perform. Likewise when we talk about synapses, you can set the synapse weight with a curve that is called Hebbian learning. But when you look at underlying data, there's lots of scatter. It's essentially impossible to set a specific synapse to a specific weight in a short period of time.

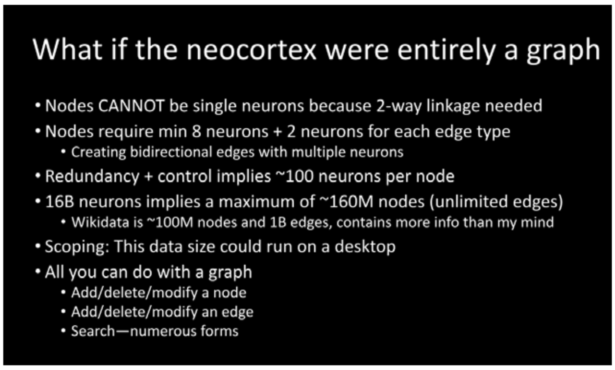
The neuron's slowness and synapse weight problems mean you cannot represent many distinct values in biological neurons. If you want a neuron's firing rate to represent a value between 0 and 10, you need some amount of time, perhaps 40ms, to represent that value. The more different values you want to represent, the slower your system is going to run. And because the neuron is slow to start with, representing 100 different values is going to make it uselessly slower. Because of that, machine learning is not particularly viable in biological neurons.



What neurons are really good at

- Graphs
  - Nodes + edges
- Examples of graphs
  - Knowledge Graph / semantic network
  - Neural network (ANN)
  - Brain
  - Computer program
- Graph data in a brain
  - Yellow is a color, blue is a color, name some colors
  - Foo is a color, bar is a color, NOW name some colors

Fig. 1: What neurons are really good at



What if the neocortex were entirely a graph

- Nodes CANNOT be single neurons because 2-way linkage needed
- Nodes require min 8 neurons + 2 neurons for each edge type
  - Creating bidirectional edges with multiple neurons
- Redundancy + control implies ~100 neurons per node
- 16B neurons implies a maximum of ~160M nodes (unlimited edges)
  - Wikidata is ~100M nodes and 1B edges, contains more info than my mind
- Scoping: This data size could run on a desktop
- All you can do with a graph
  - Add/delete/modify a node
  - Add/delete/modify an edge
  - Search—numerous forms

Fig. 2: What if the neocortex were a graph?

This leads me to graphs – collections of nodes connected by edges. Neuron clusters can do a great job of representing a graph, but individual neurons cannot because the edges of the graph must be able to be traversed in both directions while individual synapses are unidirectional. That means that edges of the graph cannot be individual synapses and it takes additional neurons in the cluster to ensure that the correct edge direction is being traversed.

We know there has to be some amount of data area in your brain that is a graph because you can know that yellow is a color and blue is a color, and from that you can be asked to name some colors and respond with yellow and blue. I can also tell you that foo is a color and bar is a color, and now you can name some colors and you can say yellow, foo, and bar. You can train your mind with a single instance of information and you know the reverse of that information instantly. That tells me there's got to be a graph (**figure 1**).

So the question is not whether or not there's a graph in your brain, but whether it's a small graph and a lot of other stuff or a big graph and a little bit of other stuff.

Suppose your entire neocortex was a graph (**figure 2**). You go to the neuron simulator and say this is what it would take to build graph nodes. At minimum, I could build a graph with eight neurons per node. You can't use one neuron per node because synapses are one directional, but you must be able to traverse this information bi-directionally for it to be useful. So the minimum I could come up with was eight neurons per node, plus two additional neurons for every edge type you want. I only programmed up two edge types, but you can add as many as you want.

A problem with this design is that the failure of any single neuron or synapse will cause the system to fail. We know the brain is highly redundant and highly capable of surviving failures. Neurons fail all the time and they don't seem to bother you very much. This implies that it's about a hundred neurons per node because that gives you redundancy and the ability to track how recently a node was used so you know when to forget.

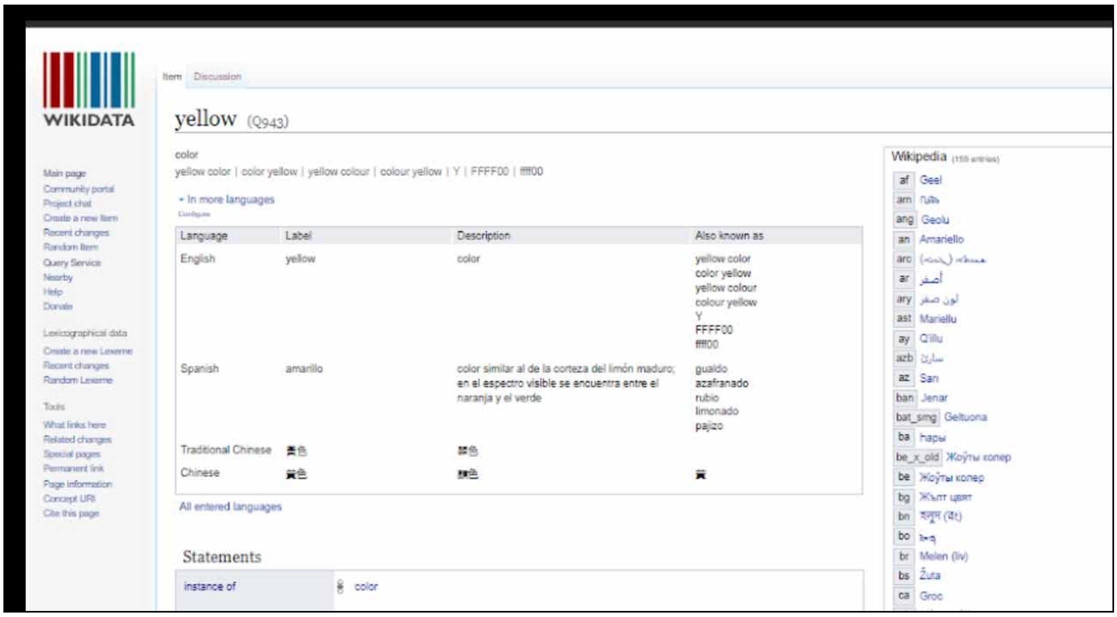
Doing a bit of division, if there are only 16 billion neurons in your neocortex, you have a maximum of 160 million nodes. That may sound like a lot, but Wikidata is a knowledge graph and it has about 100 million nodes and a billion edges and contains a ton more information than my brain. If we're talking about a system of 160 million nodes, I can put that on a desktop computer. You can add, delete, and modify nodes and edges. You can search it millions of ways. In the brain, it's likely there are not very many ways you can search, but a lot of redundancy.

How Much Data Does AGI Need?

Wikidata is a "knowledge graph" which is a collection of nodes and edges representing the relationships between various pieces of information. In Wikipedia, this knowledge graph can be used to create the summary boxes which appear on many pages.

In Wikidata, there is the node for "yellow" which contains data such as the word for yellow in numerous languages and then what they call statements like "yellow is a color" (**figure 3**).

Statements can go on for pages and highlight a distinction between the knowledge graph and the information in your brain. In a knowledge graph, when you put in



WIKIDATA

Main page
Community portal
Project chat
Create a new Item
Recent changes
Random Item
Query Service
Nearby
Help
Donate

Lexicographical data
Create a new Lexeme
Recent changes
Random Lexeme

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Concept URI
Cite this page

Item Discussion

yellow (Q943)

color
yellow color | color yellow | yellow colour | colour yellow | Y | FFFF00 | ffff00

▾ In more languages

Language	Label	Description	Also known as
English	yellow	color	yellow color color yellow yellow colour colour yellow Y FFFF00 ffff00
Spanish	amarillo	color similar al de la corteza del limón maduro; en el espectro visible se encuentra entre el naranja y el verde	gualdo azafranado rubio limonado pajizo
Traditional Chinese	黃色	顏色	
Chinese	黄色	颜色	黄

All entered languages

Statements

instance of	color

Wikipedia (155 entries)

af	Geel
am	[illegible]
ang	Geolu
an	Amariello
arc	[illegible]
ar	أصفر
ary	لون صفر
ast	Mariellu
ay	Q'illu
azb	[illegible]
az	Sarı
ban	Jenar
bat_smg	Geltuona
ba	һары
be_x_old	Жоўты колер
be	Жоўты колер
bg	Жълт цвят
bn	হলুদ (রং)
bo	[illegible]
br	Melen (liv)
bs	Žuta
ca	Groc

Fig. 3: Wikidata node for "yellow"

some information, you assume it's correct and want to keep it for a long time. In your brain, there is a constant flood of incoming information. You've got to be able to store a lot of that information for a short period and then forget what proves not to be useful. So you need to have real-time information.

Here's the real bugger in your brain. The nodes and edges cannot contain any data directly. If they did, our understanding of the brain would be much easier. We'd open up the skull and there would be all of the neuron labels and the neurons flashing whenever they fire (or when they don't). It would be easy to know how the brain was laid out. But in the brain, there aren't any unicode strings, no floating point numbers, no images. It's a graph where you can only know what a node means by the other nodes it's connected to. Everything you know is only in relation to everything else you know (**figure 4**).

Let's say this image is the abstract node representing yellow, and yellow is a color where there's some sort of abstract node representing color. If you see something yellow, it will fire this abstract node of yellow and the "yellow" node has edges which connect to things that are yellow. These yellow things, such as a banana, will have numerous connections to other nodes which define other attributes of the banana.

You have a mental model where you can keep track of a couple of bananas in your immediate surroundings. These are "instance nodes" connecting to the generic abstract banana node which can add specific detail to individual bananas while incorporating the characteristics of the abstract banana.

Now let's consider the words related to the abstract concept of yellow. You might use the word yellow or amarillo or golden. You can have multiple words associated with an individual node, but the word nodes contain no data so you (or your brain) don't know what they are either. A word must connect to its pronunciation and spelling, so you can say it, hear it, read it, or write it.

When you read the word "yellow" on this page, the recognized letters fire their respective nodes. This information will percolate up and eventually fire the abstract node representing yellow. There, you finally get an inkling as to what the word means because the abstract yellow node connects to yellow things like bananas which you can imagine in your mental model and have the "understanding" of what the word you read actually means.

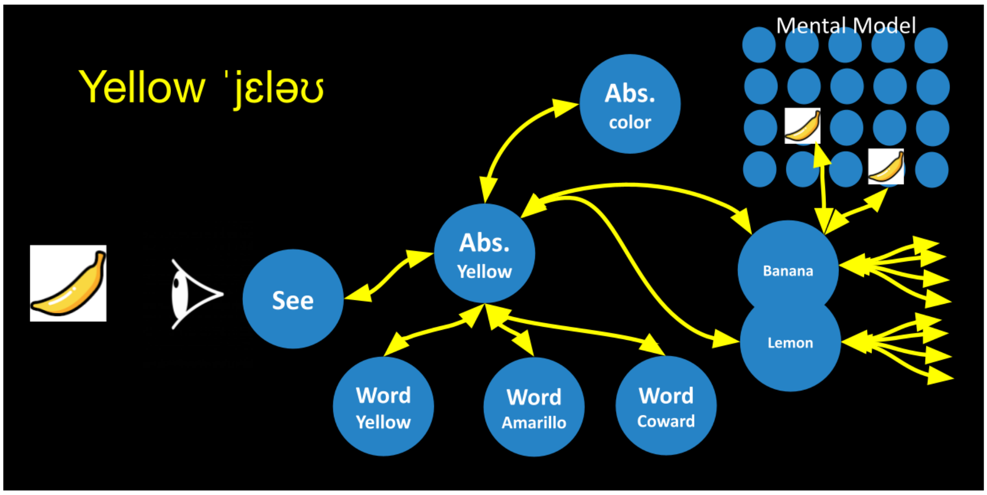


Fig. 4: Abstract node representing "yellow"

You've got to be able to run it the other way too, so that signals will pipe out to your cerebellum to coordinate hundreds of muscle contractions to say or write the word. And for each of the letters, there is an abstract node in its own right so you can speak the names of the letters to spell something aloud as well.

Your brain is a massive network of connections and you don't know what any of it means except by context – the other nodes which are connected. So you've got all of this context, and context is everything because the nodes themselves don't contain data or have labels.

It's easy to see that AGI doesn't need to follow this context-only model. We can put labels on nodes to save not only the memory needed for all those contextual nodes, but also the processing power needed to follow all of that context. There are numerous other software shortcuts available – your 3D vision, for example, which relies on merging the data from your two eyes so you can look at something and estimate in your mind how far away it is. Your brain does this with millions of neurons, but a computer can do it with a couple of lines of trigonometry which runs faster than your brain. Some robots display the kind of fluid motion with processing which essentially replaces the 56 billion neurons of your cerebellum. And it does so without modelling neurons because the programmers know about forces, physics, and feedback. They can do that in a couple of microprocessors, so the idea that you need a supercomputer to emulate all of the brain's neurons and synapses is simply unrealistic.

MLCon NEW YORK

How to write a good Training Script?

Christoph Henkelmann (DIVISIO GmbH)

While Jupyter Notebooks are a great way to do research and iterate fast, at some point comes the time where you need a robust, reliable and repeatable training process. So it is time to write your ML/DL model down in a script that can be run automatically and for much longer durations on, for example, a training server. In this talk I will present one opinionated way to write a good training script using pytorch. The talk will address a number of issues that I have encountered in practice and noticed beginners struggle with: - What needs to be configurable, what needs to be hardcoded? - How do we configure hyperparameters? - How do we make experiments repeatable? - When do we need "software architecture", when is simple copy-pasting enough? - What do we need to track and log, and how? - How to best structure your script? The goal is to give a a coarse but complete overview on how to do professional model training.

Our graph can be represented in software structures instead of neurons. That's more efficient because we've got lists and structures and hash tables. We can put labels and values in our nodes so we don't have to figure out what yellow means because we can just write yellow in the node's header and know how to spell it, say it, etc. Machine learning represents another software shortcut that may or may not have anything to do with the way your brain works, but accomplishes similar tasks. Finally, while we have a maximum of 160 million nodes, we don't know how many nodes it really takes to be generally intelligent.

Where does all this lead me? Well the amount of programming needed for AGI seems like it's going to be manageable. The size of the data is manageable and the hardware already exists to handle graphs of hundreds of millions of nodes. We've got software shortcuts that we can use as soon as we figure out what the AGI program really is. This is all to say that AGI is within our grasp as soon as we learn more about how the brain actually does its job. And with numerous scientists working on this question, the insight needed to comprehend the workings of the brain could come at any time.

AGI emergence is going to be gradual. That's because many AGI capabilities are marketable in their own right. So I produce something that has an improvement in the way your Alexa understands you and everybody's going to love that. Somebody else produces something that's got better vision that they can use on a self-driving car and everybody's going to love that. As we approach actual human level intelligence, everybody's going to love it because all of these little pieces are marketable. And the more we attach these pieces to each other and the more they can interact and have their contacts, the better things are going to be.

Finally, as we approach human level intelligence, nobody's going to notice. At some point we're going to get close to the threshold, then equal the threshold, then exceed the threshold. At some point thereafter, we're going to have machines that are obviously superior to human intelligence and people will begin to agree that yes, maybe AGI exists. But it's not going to be a specific time that happens at a specific place. My overall conclusion is that AGI is inevitable and sooner than most people think.

Charles Simon, BSEE, MSCs is a nationally recognized entrepreneur and software developer who has many years of computer experience in industry including pioneering work in AI. Mr. Simon's technical experience includes the creation of two unique Artificial Intelligence systems along with software for successful neurological test equipment. Combining AI development with biomedical nerve signal testing gives him the singular insight. He is also the author of two books - Will Computers Revolt?: Preparing for the Future of Artificial Intelligence and Brain Simulator II: The Guide for Creating Artificial General Intelligence - and the developer of Brain Simulator II.

Links & References

[1] https://futureai.guru/Founder.aspx

Controlling artificial intelligence so it behaves in our best interest

AI Alignment

At least since the arrival of ChatGPT, many people have become fearful that we are losing control over technology and that we can no longer anticipate the consequences they may have. AI Alignment deals with this problem and the technical approaches to solve it.

by Oliver Zeigermann

Two positions can be identified in the AI discourse. First, "We'll worry about that later, when the time comes" and second, "This is a problem for nerds who have no ethical values anyway". Both positions are misguided, as the problem has existed for a long time and, moreover, there are certainly ways of setting boundaries for AI. Rather, there is a lack of consensus on what those boundaries should be. AI Alignment [1] is concerned with aligning AI to desired goals. The first challenge here is to agree on these goals in the first place. The next difficulty is that it is not (yet?) possible to give these goals directly and explicitly to an AI system.

For example, Amazon developed a system several years ago that helps select suitable applicants for open positions ([2], [3]). For this, resumes of accepted and unaccepted applicants were used to train an AI system. Although they contained no explicit information about gender, male applicants were systematically preferred. We will discuss how this came about in more detail later. But first, this raises several questions: Is this desirable, or at least acceptable? And if not, how do you align the AI system so that it behaves as you want it to? In other words, how do you successfully engage in AI alignment?

For some people, AI Alignment is an issue that will become more important in the future when machines are so intelligent and powerful that they might think the world would be better without humans [4]. Nuclear war provoked by supervillains is mentioned as another possibility of AI's fatal importance. Whether these fears could ever become realistic remains speculation.

The claims being discussed as part of the EU's emerging AI regulation are more realistic. Depending on what risk is realistically inherent in an AI system, different regulations may be applied here. This is shown in **Figure 1**, which is based on a presentation for the EU [5]. Four ranges from "no risk" to "unacceptable risk" are distinguished. In this context, a system with no significant risk only has the recommendation of a "Code of Conduct",

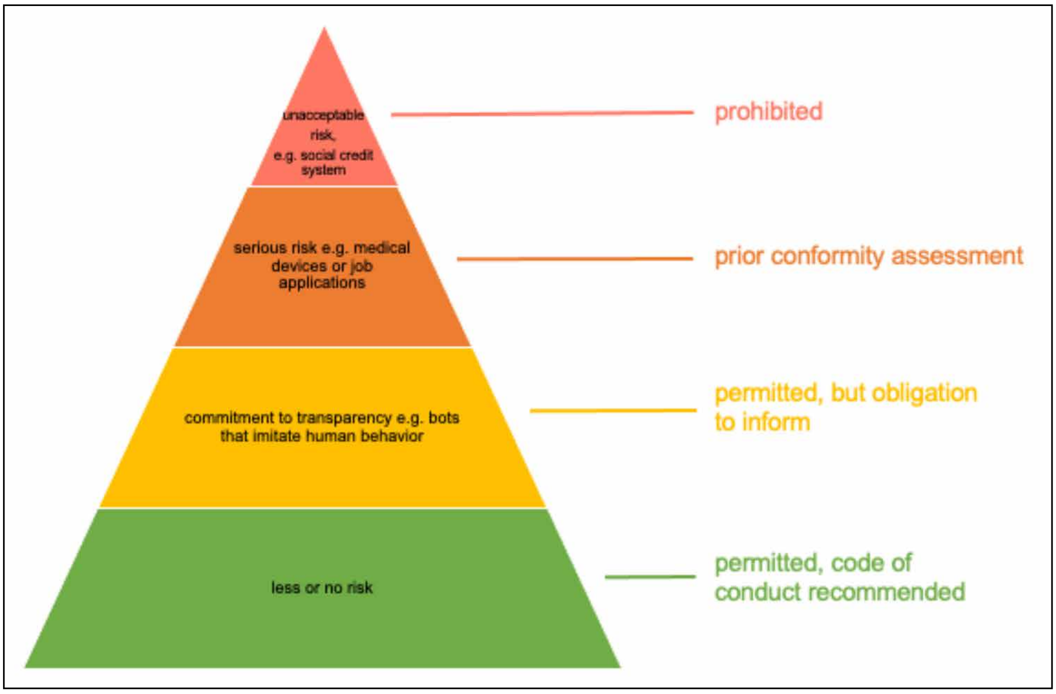

unacceptable risk, e.g. social credit system
prohibited
serious risk e.g. medical devices or job applications
prior conformity assessment
commitment to transparency e.g. bots that imitate human behavior
permitted, but obligation to inform
less or no risk
permitted, code of conduct recommended

Fig. 1: Regulation based on outgoing risk, adapted from [5]

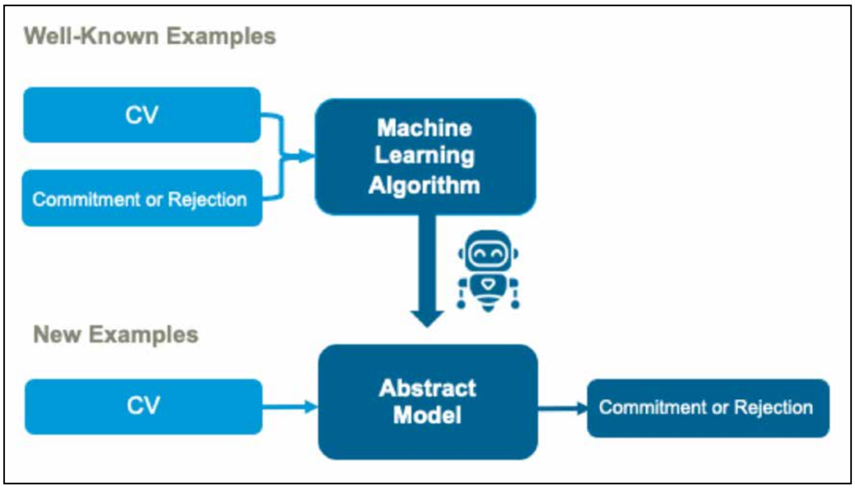


Fig. 2: How to learn from examples, also known as supervised learning

while a social credit system, as applied in China [6], is simply not allowed. However, this scheme only comes into effect if there is no specific law.

Alignment in Machine Learning Systems

A machine learning system is trained using sample data. It learns to mimic this sample data. In the best and most desirable case, the system can generalize beyond this sample data and recognizes an abstract pattern behind it. If this succeeds, the system can also react meaningfully to data that it has never seen before. Only then can we speak of learning or even a kind of understanding that goes beyond memorization.

This also happened in the example of Amazon's applicant selection, as shown in a simplified form in **Figure 2**.

Here is another example. We use images of dogs and cats as sample data for a system, training it to distinguish between them. In the best case, after training, the system also recognizes cats that are not contained in the training data set. It has learned an abstract pattern of cats, which is still based on the given training data, however.

Therefore, this system can only reproduce what already exists. It is descriptive or representative, but hardly normative. In the Amazon example, it replicates past decisions. These decisions seemed to be that men simply had a better chance of being accepted. So, at least the abstract model would be accurate. Alternatively, perhaps there were just more examples of male applicants, or some other unfortunate circumstance caused the abstract model not to be a good generalization of the example data.

At its best, however, such an approach is analytical in nature. It shows the patterns of our sample data and their backgrounds, meaning that men performed better on job applications. If that matches our desired orientation, there is no further problem. But what if it doesn't? That's what we're assuming and Amazon was of that opinion as well, since they scrapped the system.

Pre-assumptions, aka: Priors

How to provide a machine learning system additional information about our desired alignment in addition to sample data has been commonly understood for a long time. This is used to provide world or domain knowledge to the system to guide and potentially simplify or accelerate training. You support the learning process by specifying which domain to look for abstract patterns in the data. Therefore, a good abstract pattern can be learned even if the sample data describes it inadequately. In machine learning, data being an inadequate description of the desired abstract model is the rule, rather than the exception. Yann LeCun, a celebrity on the scene, vividly elaborates on this in a Twitter thread [7].

This kind of previous assumption is also called a prior. An illustrative example of a prior is linearity. As an explanation, let's take another application example. For car insurance, estimating accident risk is crucial. For an estimation, characteristics of the drivers and vehicles to be insured are collected. These characteristics are correlated with existing data on accident frequency in a machine-learning model. The method used for this is called supervised learning, and it is the same as described above.

For this purpose, let us assume that the accident frequency increases linearly with increased distance driven. The more one drives, the more accidents occur. This domain knowledge can be incorporated into the training process. This way, you can hope for a simpler model and potentially even less complex training. In the simplest case, linear regression [8] can be used here, which

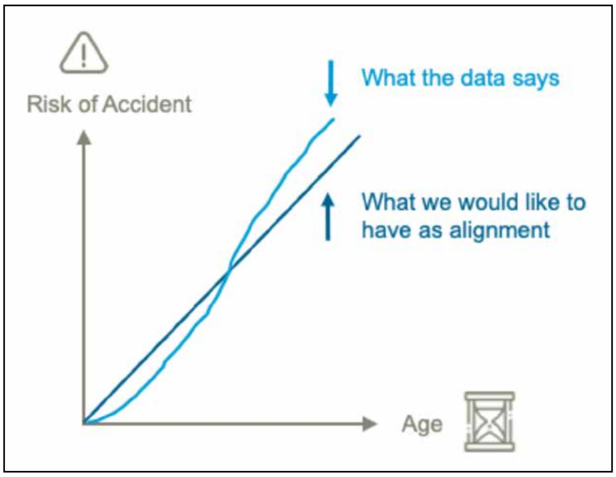


Fig. 3: Normative alignment of training outcomes

produces a reasonable model even with little training data or effort. Essentially, training consists of choosing the parameters for a straight line, slope, and displacement, to best fit the training data. Because of its simplicity, the advantage of this model is its good explainability and low resource requirement. This is because a linear relationship, "one-to-one", is intellectually easy, and a straight-line equation can be calculated on a modern computer with extremely little effort.

However, it is also possible to describe the pattern contained in the training data and correct it normatively. For this, let us assume that the relationship between age and driving ability is clearly over-linear. Driving ability does not decline in proportion to age, but at a much faster rate. Or, to put it another way, the risk of accidents increases disproportionately with age. That's how it is in the world, and that's what the data reflects. Let's assume that we don't want to give up on this important influence completely. However, we equally want to avoid excessive age discrimination. Therefore, we decide to allow a linear dependence at most. We can support the model and align it with our needs. This relationship is illustrated in **Figure 3**. The simplest way to implement this is the aforementioned linear regression.

Now, you could also argue that models usually have not only one input, but many, which act in combination on the prediction. Moreover, in our example, the linear relationship between distance driven and accident frequency does not need to be immediately plausible. Don't drivers with little driving experience have a higher risk? In that case, you could imagine a partial linear relationship. In the beginning, the risk decreases in relation to the distance driven, but then it increases again after a certain point and remains linear. There are also tools for these kinds of complex correlations. In the deep learning field, TensorFlow Lattice [9] offers the possibility of specifying a separate set of alignments for each individual influencing factor. This is also possible in a nonlinear or only partially linear way.

In addition to these relatively simple methods, there are other ways to influence. These include the learning algorithms you choose, the sample data selected, and, especially in deep learning, the neural network's architecture and learning approach. These interventions in the training process are technically challenging and must be performed sparingly under supervision. Depending on the training data, otherwise, it may become impossible to train a good model with the desired priors.

Is all this not enough? Causal Inference

The field of classical machine learning is often accused of falling short. People say that these techniques are suitable for fitting straight lines and curves to sample data, but not for producing intelligent systems that behave as we want them to. In a Twitter thread by Pedro Domingos [10], typical representatives of a more radical course such as Gary Marcus and Judea Pearl also come forward. They agree that without modeling causality (Causal Inference), there will be no really intelligent system or AI Alignment.

In general, this movement can be accused of criticizing existing approaches but not having any executable systems to show for themselves. Nevertheless, Causal Inference has been a hyped topic for a while now and you should at least be aware of this critical position.

ChatGPT, or why 2023 is a special year for AI and AI Alignment.

Regardless of whether someone welcomes current developments in AI or is more fearful or dismissive of them, one thing seems certain: 2023 will be a special year in the history of AI. For the first time, an AI-based system, ChatGPT [11], managed to create a veritable boom of enthusiasm among a broad mass of the population. ChatGPT is a kind of chatbot that you can converse about any topic with, and not just in English. There are further articles for a general introduction to ChatGPT.

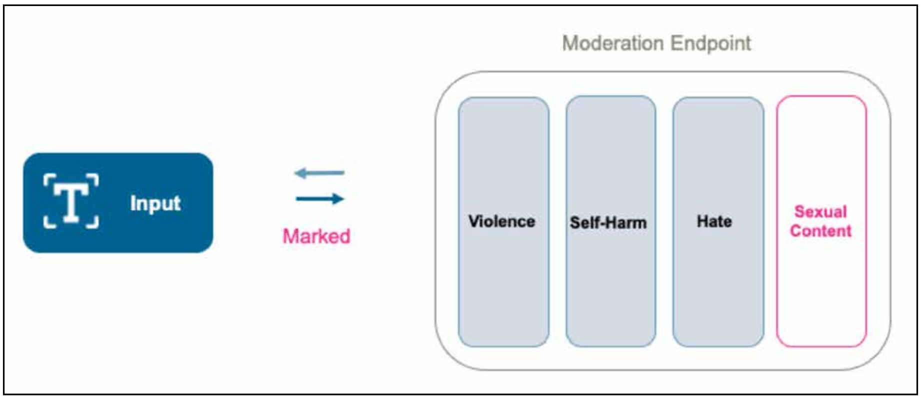


Fig. 4: A moderation system filters out undesirable categories

ChatGPT is simply the most prominent example of a variety of systems already in use in many places. They all share the same challenge: how do we ensure that the system does not issue inappropriate responses? One obvious approach is to check each response from the system for appropriateness. To do this, we can train a system using sample data. This data consists of pairs of texts and a categorization of whether they match our alignment or not. Operating this kind of system is shown in **Figure 4**. OpenAI, the producer of ChatGPT, offers this functionality already trained and directly usable as an API [12].

This approach can be applied to any AI setting. The system's output is not directly returned, but first checked for your desired alignment. When in doubt, a new output can be generated by the same system, another system can be consulted, or the output can be denied completely. ChatGPT is a system that works with probabilities and is able to give any number of different answers to the same input. Most AI systems cannot do this and must choose one of the other options.

As mentioned at the beginning, we as a society still need to clarify which systems we consider risky. Where do we want to demand transparency or even regulation? Technically, this is already possible for a system like ChatGPT by inserting a kind of watermark [13] into generated text. This works by selecting words from a restricted list and assuming that a human making this specific combination has an extremely low probability. This can be used to establish the machine as the author. Additionally, the risk of plagiarism is greatly reduced because the machine - imperceivable to us - does not write exactly like a human. In fact, OpenAI is considering using these watermarks in ChatGPT [14]. There are other methods that work without watermarks to find out whether a text comes from a particular language model [15]. This only requires access to the model under suspicion. The obvious weakness is knowing or guessing the model under suspicion.

Conclusion

As AI systems become more intelligent, the areas where they can be used become more important and therefore, riskier. On the one hand, this is an issue that affects us directly today. On the other hand, an AI that wipes out humanity is just material for a science fiction movie.

However, targeting these systems for specific goals can only be achieved indirectly. This is done by selecting sample data and Priors that are introduced into these systems. Therefore, it may also be useful to subject the system's results to further scrutiny. These are issues that are already being discussed at both the policy and technical levels. Neither group, those who see AI as a huge problem, and those who think no one cares, are correct.

Oliver Zeigermann is a software developer and architect from Hamburg, Germany. He has spent the last 40 years developing software with different approaches, hardware and programming languages. He works as head of AI and AI strategist at open knowledge GmbH.

Links & References

[1] https://en.wikipedia.org/wiki/AI_alignment

[2] https://www.bbc.com/news/technology-45809919

[3] https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G

[4] https://www.derstandard.de/story/2000142763807/chatgpt-so-koennte-kuenstliche-intelligenz-die-menschheit-ausloeschen

[5] https://www.ceps.eu/wp-content/uploads/2021/04/AI-Presentation-CEPS-Webinar-L.-Sioli-23.4.21.pdf

[6] https://en.wikipedia.org/wiki/Social_Credit_System

[7] https://twitter.com/ylecun/status/1591463668612730880?t=eyUG-2osacHHE3fDMDgO3g

[8] https://en.wikipedia.org/wiki/Linear_regression

[9] https://www.tensorflow.org/lattice/overview

[10] https://twitter.com/pmddomingos/status/1576665689326116864

[11] https://openai.com/blog/chatgpt/

[12] https://openai.com/blog/new-and-improved-content-moderation-tooling/

[13] https://arxiv.org/abs/2301.10226 and https://twitter.com/tomgoldsteincs/status/1618287665006403585

[14] https://www.businessinsider.com/openai-chatgpt-ceo-sam-altman-responds-school-plagiarism-concerns-bans-2023-1

[15] https://arxiv.org/abs/2301.11305

A theoretical and practical approach towards finding it

The X factor in AI

Welcome to the world of AI. This is actually - you. This statement arouses some questions in each reader, I will be sure. About the meaning of the I, being involved in AI, about the process and consequences of stating it. This article aims in representing a perspective of the definition of AI we take as granted, but that could define the past, present, and future of the world of AI. However, as you might observe in the title, there is never just a definition, or just a statement, there is always something behind it. In our case, something in between. A factor, a secret spice, that makes AI what we would imagine it being in the future, how we see it today, and what it has been before.

by Ekaterina Marinova

The concept of artificial intelligence consists of the idea of our human intelligence being used in the space of the artificial – the world of the acceleration. Machine power, speed and memory used for something more, used as a power of our thoughts, actions, and logic. However, very often we may find that this definition, this understanding of AI is not that common, and people tend to overuse the word and underestimate the meaning. Sometimes, we even get to points of extremes of differentiating ourselves from AI fully or consider it part of our lives fully.

At a point of defining artificial intelligence each person would look at the future typically, sometimes include the present, and rarely the past. This is exactly the point in which I would like to raise a hand and ask of all of you to consider this time perspective as one thing, that AI fully incorporates within its essence. The past serves as the starting point of AI. This is the historical information and knowledge, that AI would need to survive, to develop and to future evolve. As us – people.

The present would give a different perspective. It would bring new moments, new information, new look. However, it will always be evaluated as per the historical data. The present will be measured as good or bad, as additional, or less, as new or old, always based on the past. And this is the point, which AI will use to evaluate its past, draw its future and reflect on the now. As us – people.

Finally, the future, the point in time which AI is mostly associated with. This is the thing we , as people, fear mostly. We fear- because we do not know and we cannot predict. We fear, because we consider AI being able to predict it and manipulate it. However, this is the point in which we would think , how does AI actually do that. And we would need to go back to the previous points. We cannot talk about future, without understanding the past, evaluating the present and scoring the potential of the future moments based on that. And just like that, the future will be based on everything before and will have its chances to happen. As it will with us – people.

If we take the AI chess algorithm that could defeat even grandmaster players. This cannot be a computer that thought itself out of nowhere. It should have the knowledge, it should play games to evaluate this knowledge, and then it should master – as us people. However, we cannot expect this algorithm to be able to give us an advice on swimming – because this would be something totally unfamiliar with. Every AI is about our knowledge and our future, and we should act upon that

To Do: Please, take a moment and think of your AI. Design the thought, process, activity, anything you may need the power of the machines for.

Congratulations. Now, you have the idea of your AI solution, already designed in your head.

To continue this, I might need to give a bit of a information on what is possible from now on that you have set your goals and you have the design of the solution in your head. And yes- the world of IT evolves around artificial intelligence. We develop solutions, work on top notch technologies, scenarios, processes, automations. All components of AI. All have a point of touch with the definition of AI – our past, present, and future.

The world of AI spans across galaxies of technologies, solutions, and inventions. One may perceive it as robots and self-driving cars, while to another this might be just a chatbot solution or a social media app. And both of them would be correct, based on their perceptions and knowledge. Because, as you can already deduct by yourselves AI would be representing a different aspect of our knowledge, of our experience, and would be serving different need or opportunity of our future or imagination of future.

If you are about to search online for AI technologies, you will find plenty of them. You will be lost between machine learning, neural networks, virtual companions, autonomous robotics, real-time emotion analytics, et cetera. And if I try to categorize them you would be lost, and probably I will skip some of the available technologies. This is why I have selected to give you a process and framework to be able to find these solutions for yourselves.

The technologies behind AI work in a way that lies still in the definition – past, present and future. First, when you are building your solution, you would need to have historical information to be able to learn from. Imagine in our case, we would like to build a solution that recognizes our friends and family within the city. Now, what you will need as information is how they look like – thus, you will need pictures of your family. Once you have that, you could use the available technology, enrich it with this data, automate it and look for the future outcome. In this case, we will be looking into image processing and recognition and facial recognition, as part of computer vision as well, to be able to use algorithm that would use pictures to recognize people on the street or within the city.

When considering this artificial intelligence, you will be considering a solution that would enhance your future, and outcomes that would serve you there. The present will be used to evaluate this, because as you can imagine people will grow old, pictures will change, thus your knowledge, thus the present will become the past, and the future outputs will be differing.

And this is the point, where we would say that the I and the AI would meet. Using our knowledge, feeding algorithms and machines to enable us to do more. However, still working together and requiring us – as people – to adapt, change skills, update knowledge, and help AI perform better.

As this is only one example of the available technologies, I could go further into robotics, cognitive services – recognizing emotions, facial expressions, analyzing social media data, personal assistants and so on. But it will always be the same – knowledge, technology, output, reiteration.

To Do: Let's take a moment to recap. It may looks like AI is only technologies and then again, not that simple. However, as you may have noticed – all of them have transitioned and evolved to a point where they could be used as tools by anyone – no matter the skills or industry. Now, please select and map the technologies you find mostly relevant to your idea.

Success! You have built your technology map and you are one step closer to bringing your solution to life.

Now, you would ask what happens after that. Everybody could do the idea part. Everybody can draw this design and can think of their solution, right. First, I would like to say congratulations on your progress. You have made it to here and now it only gets easier and more interesting, as we will turn to the I part in the AI or exactly - how do we get from idea to actual AI – our AI.

If it is that easy to draw a solution – then why do, we have successful AI solutions? Why are some better than the others, and some just do not make sense to us. To answer those questions, we would need to answer another question as well, what happens when our AI solution is out there and is a success? The result of our solution would depend on the effect it has out there entangled to the purpose it was given. It is more like a formula for success, that is so natural, and then again, a bit hard to understand. However, easy to evaluate from technical perspective, but gets harder when social and ethical components are taken into considerations. For example, the self-driving car. Perfectly designed technical solution, with clear purpose and goal. Success from

AI solution perspective, but a huge ethical dilemma and societal challenge.

The present give us an opportunity to think of the future and reflect on the past when designing the AI solution. Additionally, it gives us the opportunity to select current technologies, and people to work with. However, there is a risk. The risk of the outdated technology within a year, the risk of not being able to analyze the future needs very well, the risk of misunderstanding the past, and the risk of selecting people with skills that soon enough could be outdated. Therefore, we need to seek for an adaptive approach towards our solution.

The most important component is our knowledge. You cannot be a person who has never touched or seen a car to be able to design a self-driving one. In order to get there, you would either need a person who can provide this knowledge, or you may need to immerse yourself into research so deep, you could draw the car in your mind. The knowledge component is key to your solution, as this is the point in which you meet all three-time perspectives and are able to paint a picture that is unique. This is also the point where you need to be adaptive. You need to think of the additional research, future ideas, and papers out there, that may complement your solution, give it additional opportunities, or be a risk. For example, think of the autonomous smart cities and the fit the self-driving car would be.

Then think of technologies. What we consider as our comfort zone with technologies and know they could give us. And what is out there as additional, innovative approaches towards technologies. Thinking one step further of what we have, one step further of what is to be announced, and thinking of the technological direction. By doing this, you will be leaving out space for your solution to further develop and adapt, based on the newest trends on the technology world map.

And finally, the skills that you will need. This is not often directly correlated to the people you will be working with, but rather your skills, the one you need one, and those you, and the people using your solution, would be needing in the future. The skills topic is a huge one, especially when talking about the consequences of AI.

Given the example of the self-driving car, you will no longer be needing to know how to drive car with a manual gear box, or automatic. But rather, which buttons to press and what roads to select and navigate with this new technology. Aside from that, think of the ethical and societal aspect of the solution and learn how to make choices, and how to deal in different situation with this car. Now going back, you would say that your knowledge would be the underlaying one, but adding the research perspective, the technological opportunities and future perspective, this could alter to a point, that you may need to reskill yourself to the solution you have created.

To Do: Now, please, take a moment to think of what would you need as skills, people, research? Is your knowledge enough, could you gather more information and data points to enrich it? When you structure this overview, you would find yourself understanding the solution better and being able to color your design.

Voila! Looks like you have a plan – overview, resources, and structure.

However, many people do have ideas and technologies available, knowledge and research, skills and perspective of the future. However, their solution is still not there. It cannot get to the market, or cannot realize it potential, or cannot meet the objectives set at the very beginning. So you end up there, building your equations, designing and researching for the best solution, but there is still something missing – the thing I will call the X factor in the AI formula that each and every one of us has to be able to understand and implement.

This is data. Data is usually perceived as something very technical, often connected with just databases and SQL. Sometimes, used in a bit more than that – like data science and preparation. However, stepping aside from the technical perspective and stepping into the knowledge and wisdom one , would give you the ability to really utilize its power, and use it as the X factor of your solution. Being able to understand data, and work with data would give you the ability to go through each step of the solution and the process in a data-driven way, not just based on ideas – but rather hypothesis. Not just

using technology but evaluating it. And finally, not just expecting output – but testing it.

Your idea data path would often relate to your knowledge or the research you have selected to do. However, it will help you build your hypothesis. The questions you want to answer by creating these solutions. And those will be the questions you would get back to, when evaluating it. Data about the idea and the universe– where are we working, what do we want to do with this solution, is it really needed, is this the medicine, rather than the candy for the world where the solution will be released.. AI technology is about bringing value, so data about this value and output needs to be accumulated and evaluated. Calculating the effect, calculating the efforts, resources and investment from human, time and material perspective. Calculating what would you get at the end as in time, emotions, or money.

You would expect to select a technology that already considers this data components. And you will be right, all of them are based on that. But in every solution, you will have your first point of selecting your data. Choosing your data sources and thinking of what it is required, and whether you have it or not. It is always data in the core of all AI solutions, as it is the machine-readable format of our human intelligence.

And finally, evaluate our results. Is our solution serving its purpose? Did we go one step further, or do we need to restart? Thinking of the questions we have asked, and the knowledge we had in the point of beginning the solution and the point of releasing it out there. Thinking of the future of this result, and always not being a final one. Thinking of the opportunities that data would give us, we would be able to go back the data path and enhance, enrich, and update the solution to a part where it makes it even more successful, or design it to a part, where it is different, and could even answer a different question.

This is the point, where you could expect something totally unexpectable. And this will be alright. As environment and surrounding factor change thus the AI would change. If the I part varies, then the whole AI would vary as well.

Data is the language that speaks between the two worlds – the artificial and the human intelligence ones, and it will adapt, change and evolve, as our solutions will.

To Do: Draw you data path. Draw your knowledge data path, technology data path and output data path.

Congratulations! Now you have it all. Enable your skills, find the technologies, dive deeper into the research and knowledge you have been working with, and focus on what you want to create.

Rethinking what we know is a kick out of our comfort zone as people. The purpose of doing that would be to grow as individuals, to explore new worlds, or redesign the one we live in – make it better by looking back to the past, evaluating the present and designing the future. It takes huge companies, small startups, and single people to do that. So it would be everyone thinking of using our knowledge and skills to move forward. There is an X factor in everything that we do, and we need to think about it, find it and master it.

Think of your knowledge.
Think of your present.
Design your future.
This is the power of data.
This is the power of AI.

Ekaterina Marinova is a Senior Manager for Digital Banking and Innovation at Raiffeisenbank Bulgaria. She is also an adjunct lecturer on big data and machine learning at AUBG. In addition, she is completing her Ph.D. thesis on data philanthropy as an innovation accelerator for emerging markets in the context of smart cities. As a Microsoft MVP in AI, she participates in the Microsoft community as a speaker, tech event organiser, and community educator.

A more flexible approach to ML

The Future of Machine Learning is Liquid

Machine learning is a rapidly evolving field that has already transformed the way we interact with technology. However, as machine learning continues to evolve, new concepts and methods are being developed that have the potential to revolutionize the field even further.

by Nahla Davies

One such concept is "liquid machine learning", which is a novel approach [1] to machine learning that involves training models on constantly changing, real-time data streams. As with any new concept, however, there are also significant challenges associated with implementing liquid machine learning. These include technical challenges, such as developing algorithms that can learn from constantly changing data, as well as ethical concerns surrounding issues such as data privacy and bias.

Nonetheless, the potential benefits of liquid machine learning are too significant to ignore, and as such, it is a topic that is generating increasing interest among researchers and practitioners alike. In this article, we will explore what this new technology is and the features that make it promising.

What is liquid machine learning?

Liquid machine learning models are designed to be highly adaptive, with the ability to learn from changes in the environment, user behavior, or other contextual factors that may influence the data.

Unlike traditional machine learning methods, which rely on static datasets [2], liquid machine learning models are designed to continually adapt and learn from new data, making them highly flexible and capable of handling complex and dynamic problems. This approach has the potential to transform many industries, from finance and healthcare to transportation and logistics, by enabling real-time decision-making that is based on the latest data.

The key difference between traditional machine learning and liquid machine learning is that the latter enables models to be trained on dynamic, real-time data streams. This allows the model to continually adapt to new and evolving data, as opposed to static datasets that are often limited in scope and can become outdated quickly.

Liquid neural networks were first defined by a team [3] of researchers at MIT led by Ramin Hasani and Mathias Lechner. The first research and paper were published in 2020, with the revised and more advanced coming in November 2022.

Machine learning and deep learning algorithms have existed for decades, but they all have one drawback. They can only learn a fixed mapping from input to output based on the training data they receive. This made them vulnerable to changes in the environment around them.

To address this hiccup, developers often embrace active learning techniques that continuously re-label, re-train, and re-deploy algorithms. This can be daunting. Liquid machine learning solves this using its unique features.

So what features do liquid machine learning neutral possess that make them more advanced than the traditional ones?

Key features: Adaptability

Unlike traditional neural networks, which have fixed algorithms set during training, liquid neural nets can adjust their underlying equations based on the input they observe. This adaptability allows liquid neural nets to handle real-world data better and provide more accurate results. They can acknowledge patterns by inspecting training examples and adjusting their weights and parameters.

The inspiration for these neural networks comes from the microscopic nematode C. elegans, which has a mapped-out nervous system. By mimicking the processing pathways of the brain, liquid neural nets can achieve greater accuracy in recognizing and classifying complex patterns. Like the worms they are based on, liquid networks can instantly adapt and transform their underlying algorithms.

Liquid neural networks are more compact. Early experiments show that these networks can run faster and more accurately [4] than continuous-time neural networks, which model systems that vary over time. This is because liquid neural networks can continuously analyze success metrics and incorporate new information to improve performance over time.

The ability to continuously adapt to new data inputs means that liquid networks can have many useful real-world applications. For instance, the breakthrough may benefit decision-making processes that rely on dynamic data streams, such as medical diagnoses and self-driving cars.

Responsiveness

Typically, machine learning networks can solve problems or decode messages by adhering to rules and regulations. However, when it comes to recognizing speech patterns, imaging, and medical diagnoses, more variables that can adapt and be responsive come to play. Traditional neural networks can struggle with this.

Liquid neural networks exhibit greater adaptability [5] as they can modify underlying equations based on the input they observe in real-time. In particular, they can alter the responsiveness of neurons, i.e., how rapidly they react. This network possesses remarkable proficiency in resolving complex problems and can even surpass the capabilities of humans. And they achieve this without additional training. Liquid machine learning networks demonstrate responsiveness and accuracy in tasks such as identifying human activities from motion sensors and simulating the physical movements of a walking robot. Notably, in a sample of 8,000 patients [6] for medical predictions, the new version of the liquid neural network called "closed-form continuous-time" (CfC) outperformed other models by a significant factor of 220.

The responsiveness of liquid machine learning can also play a critical role in fraud prevention in banking especially given their adaptive ability. They can be used in tandem with other self-learning technologies to identify fraudulent activity by analyzing patterns and anomalies in a system. This will reduce the overreliance on FDIC and its limitations. Currently, consumers have to look for bank accounts that come with FDIC insurance, meaning that they'll be covered up to $250,000 [7] in the event of a breach or default.

Affordability and scalability

One of the biggest advantages liquid neural networks offer is affordability and scalability.

By focusing on fewer but richer nodes in the algorithm, liquid networks can scale down the network, resulting in lower computational costs. This reduction in computational costs is achieved by relying on fewer but more sophisticated computing nodes, making the neural network more affordable to implement.

In addition to their scalability, liquid neural networks have smaller and fewer computing nodes [8] than traditional neural networks. This makes the network more affordable and easier to implement, as fewer computing resources are required to run the network. The few but richer nodes make this possible.

Transparency

Liquid machine learning is a breakthrough technology that addresses the limitations of traditional neural networks.

Traditional neural networks had a major drawback; they struggled to diagnose why the network made a certain characterization. Liquid-neutral networks can give developers insight into a system's decision-making process, so they can easily identify and fix issues in the system.

In simpler terms, liquid networks make algorithms more interpretable. It achieves this by overcoming the "black box" conundrum [9], a major concern for engineers who want to understand why an AI system comes to certain conclusions. This technology allows developers to understand the reasoning behind the algorithm's decision-making process. The inherent flexibility of liquid neural networks renders them applicable to virtually every technology available today. Its seamless integration allows individuals to utilize it in any way they desire. Greater flexibility makes models more robust by improving their ability to withstand unexpected and noisy data. For instance, during trials, Hasani and his team at MIT's Computer Science and Artificial Intelligence Lab discovered that the network could enable algorithms to adapt to heavy rains that obscure a self-driving car's vision.

Such advances can improve autonomous driving and lower the number of traffic offenses that can lead to hikes in insurance premiums. In Canada, for instance, the first offense of careless driving can cause your insurance premiums to increase by 100% [10], while in the United States, it can increase by up to $1,130 [11].

What the future holds for liquid machine learning

The liquid neural network is a promising programming paradigm that has become a reality. It replaces the traditional approach of precisely defining a task for the computer with a more adaptable and intuitive system, enabling computers to solve problems with less precise information. The emergence of liquid neural networks [12], drawing inspiration from small species' brains, promises to revolutionize machine learning. It plans to achieve this by providing flexible and robust models that adapt to changing conditions.

These networks have a broad range of real-world applications, particularly in safety-critical tasks like driving and flying, where they can enhance decision-making capabilities. Another area where machine learning is poised to make a big impact is healthcare. By analyzing patient data and medical records, machine learning algorithms can identify patterns and make predictions about patient outcomes. This can lead to more personalized and effective treatments, as well as better disease prevention and management.

Machine learning also has the potential to revolutionize [13] the way we interact with technology. As machines become better at understanding natural language, they will be able to provide more intuitive and personalized experiences for users. For example, virtual assistants such as Siri and Alexa will be able to understand complex requests and respond more intelligently.

However, as machine learning continues to evolve, there are also concerns about its potential impact on society. For example, there are concerns about the potential for bias in machine learning algorithms, as well as the ethical implications of automating tasks that were previously done by humans.

Despite these concerns, the potential benefits of machine learning are too significant to ignore. As we continue to develop new approaches such as liquid machine learning, we will unlock even more potential in this exciting field.

Conclusion

Overall, it's clear that machine learning is a field with enormous potential to transform our lives. The emergence of new approaches such as liquid machine learning is particularly exciting, as it enables real-time decision-making based on constantly changing data streams.

As machine learning continues to evolve, we will undoubtedly see even more innovative and transformative applications emerge, from healthcare to transportation and beyond. However, it's important to address the challenges associated with this technology, including issues around bias and ethical concerns. By doing so, we can ensure that machine learning is developed responsibly and equitably [14], which benefits society as a whole.

Nahla Davies is a software developer and tech writer. Before devoting her work full time to technical writing, she managed — among other intriguing things — to serve as a lead programmer at an Inc. 5,000 experiential branding organization whose clients include Samsung, Time Warner, Netflix, and Sony.

Links & References

[1] https://www.quantamagazine.org/researchers-discover-a-more-flexible-approach-to-machine-learning-20230207/

[2] https://www.geeksforgeeks.org/static-data-structure-vs-dynamic-data-structure/

[3] https://news.mit.edu/2021/machine-learning-adapts-0128

[4] https://devm.io/machine-learning/basic-introduction-machine-learning-145140-001

[5] https://www.analyticsinsight.net/why-is-liquid-neural-network-from-mit-a-revolutionary-innovation/

[6] https://interestingengineering.com/innovation/mit-reveals-a-new-type-of-faster-ai-algorithm-for-solving-a-complex-equation

[7] https://www.creditdonkey.com/online-savings-account.html

[8] https://techcrunch.com/2021/01/28/mit-researchers-develop-a-new-liquid-neural-network-thats-better-at-adapting-to-new-info/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAABXpZhuR-RkoKmozNG-5MymK_LWqhxx9csGQMRpbLQIAFwkC3ZrRVlRKYJu4rQ39cRXYpwb3IL-UOeyDKsRgnCTyInbe1DDhnsG9mtGxlaXbTg9NSkHqIt837BuG77QqqioKeHKF0lZLD0PEB1nHnaTRkBdKA62vDnPyqr_GAs5h

[9] https://www.linkedin.com/pulse/ai-blackbox-conundrum-gavin-wickremasuriya

[10] https://www.mychoice.ca/blog/careless-driving-in-ontario-points-fines-faqs/

[11] https://www.thebalancemoney.com/careless-driving-insurance-rate-527495#:~:text=A%20citation%20for%20careless%20driving,prevention%20is%20the%20most%20effective.

[12] https://blog.roboflow.com/liquid-neural-netowrks/

[13] https://www.pluralsight.com/guides/machine-learning:-a-technological-revolution-of-the-future

[14] https://devm.io/machine-learning/develop-ml-responsibly-146690-001

What does the new model bring?

The Transition from GPT-3 to GPT-4: A Step Forward

OpenAI recently released GPT-4, their newest version of the GPT model, which is also used in ChatGPT. We discussed GPT-4 in depth with AI specialist Pieter Buteneers, including its fine-tuning process, capabilities, the Microsoft-OpenAI relationship, and the importance of open-source in the field of AI.

devmio: Hi Pieter, thank you for taking the time to do this interview! Could you please introduce yourself to our readers?

Pieter Buteneers: My name is Pieter Buteneers, and I am currently the director of Sinch Labs at Sinch, but only for one more week. I launched my own startup in November, and because it's really taking off, I can't combine the two jobs. So, starting in April, I'll just be the CTO of transfo.energy [1], a company that seeks to match supply and demand in the energy market as closely as possible.

devmio: Could you explain to our readers what GPT-4 is? Have you had a chance to test it yet?

Pieter Buteneers: Well, I have tried it out, but not directly. I have tried out Bing Chat, which is already supported by GPT-4. The difference between ChatGPT and Bing Chat is that Bing Chat also provides internet search results as answers. It actually searches the internet before giving you an answer, and it bases its answer on what it finds there as ground truth. So you cannot really compare Bing Chat to standalone GPT-4. GPT-4, in and of itself, is just something that completes your sentence. So, if you stop somewhere halfway through your text, it will continue that text until infinity. That's what GPT-3 did before it, and GPT-2 before that, and so on.

So it's simply a large language model that OpenAI trained to predict what would happen next. It determines what comes after the last words or sub-words, called tokens, and then fills in what's next. You can keep doing this until you get whole sentences, which is what GPT-4 is.

The only significant difference between GPT-4 and GPT-3 or GPT-3.5 is that, at least according to OpenAI, GPT-4 can also process information obtained from images. It can extract useful information from both an image and a text and merge it into a single set of information to continue a conversation. That's why it was so effective at answering exam questions. In some cases, an image or a drawing was included in an exam, which is particularly important for producing the right answer.

devmio: How much do we know about the new model?

Pieter Buteneers: The main issue is that, with GPT-4, OpenAI has become even more closed-source than it was with GPT-3. They at least provided a paper outlining how various things worked with GPT-3. The same thing happened with DALL-E, where they had some papers explaining how it worked before other people jumped on it and created something like stable diffusion, which is an improvement over DALL-E.

I think they are super afraid that if they disclose too much information, the open-source community will try to replicate what they have. Worry, I believe, stems from a fear of lost income rather than a fear of misuse.

devmio: What's going on behind the hood is difficult to guess. Do you have an idea how much bigger the model is in comparison to the GPT-3?

Pieter Buteneers: Well, the assumption I came across on the internet for the language model part is that it's not necessarily that much bigger than before; they just trained it for a much longer period of time. According to the most recent papers on LLMs, training a language model for longer and longer periods of time really improves its performance with the same number of parameters and the same size.

They most likely use transformers for image recognition, and my guess is that they do something similar to what they do for text: During the training process, parts of the image are removed and the model tries to recreate what is missing in the image. The model is then used to generate some kind of embedding, which describes what the image actually is. Because you must understand fully what is depicted in the image if you are to complete the missing information. In the training process, parts of the image are removed and the model tries to reconstruct the missing parts of the image. And then you use this model to create a kind of embedding, something that describes what the image actually contains. But we can only speculate. It's very frustrating for me as a machine learning engineer who enjoys tinkering with these things that they don't disclose much information on how GPT-4 works.

devmio: So GPT-4 doesn't have a lot more parameters than GPT-3? What does the model's parameter number reveal about it?

Pieter Buteneers: So, up until GPT-3, the trend was that every time you increased the size of your model, it became much more powerful. As a result, everybody assumed they would simply increase its size. People presumed that because GPT-3 had 175 billion parameters, they would expand five or ten times the size, to a trillion parameters or something like that.

If you have 175 billion parameters, like in GPT-3, this means that all of GPT-3 or GPT-3.5's knowledge is stored in just 175 billion numbers. That's all there is to it. It only contains that much information.

Many papers show that you can take a large network and reduce over half of the parameters while still getting a fairly comparable result to the original network. So there is a lot of redundancy in this network, and many of the parameters contribute in the same way and don't add much to the model, so you can shrink it. This leads to the assumption that there is still room to learn more within the model. So that's why I think that by training models over a longer period of time, you may find ways to force more information into them.

devmio: What does it mean to train the model longer in that context? Is it giving the model the same texts over and over again? Or different texts and more reinforcement learning?

Pieter Buteneers: Basically, you give it a piece of text, whether it's a sentence, half of a sentence, or even half of a book. It doesn't really matter. Then you say, "Tell me what comes next." And the interesting thing is you can use any text you find on the internet. Higher-quality text will obviously produce better results, but in practice, any text will suffice. So training it longer can mean one of two things: give it more text to train on or have it finish more sentences and so on. Yet we have no idea what OpenAI did.

Nonetheless, it's not that bad to re-feed the data set a few times and ensure that the model is capable of extracting more information from the text by iterating again. You don't teach the network exactly what to say; instead, you let it take a small step in the right direction with each iteration. We've discovered that taking smaller steps results in a more stable network that can learn more broadly. They have shown that if you follow this step-by-step approach for a longer period of time, you can get better results with substantially fewer parameters.

devmio: So this is what OpenAI most likely did to upgrade from GPT-3 to GPT-4?

Pieter Buteneers: They undoubtedly trained the model longer because it is what is discussed in numerous research papers from Google, Facebook, and others. It's very likely that they will do so, and it's also possible that they have discovered another way to use, perhaps a trillion or a few trillion parameters. It's pure guesswork, but it's the most educated guess we have so far. It's only a minor improvement over GPT-3.

devmio: On their blog, OpenAI wrote that they reworked the alignment of GPT-4. What do they mean by that?

Pieter Buteneers: If GPT-4 has been trained on the internet, it will reproduce whatever it finds there. It's not always super useful, and because it tries to learn the most common patterns, it can be very misogynistic or racist. So that's why there is always the risk that these models will learn the simplest patterns first.

They did what they had done for ChatGPT, which was to let it produce results and then rate them using a different model. And, using that rating model, you can use reinforcement learning to improve the output, similar to how ChatGPT was trained. It's actually much easier to build a model that ranks answers as "good," "mediocre," or "bad" than it is to come up with a good answer. Just like it's far easier to be a critic than it is to create something.

It's the same with machine learning models: OpenAI created one for GPT-3.5 to turn it into ChatGPT, with people labeling data and rating answers. Once you've created this model, you can apply it to the output of any text generation model and use it as feedback. That's where reinforcement learning comes in, since you simply state "good" or "bad" in the feedback, and the model works out why.

To improve the output, you need a lot of iterations. So, that takes a significant amount of time. It's difficult to tell what they really did under the hood. But, once again, my best guess is that they used reinforcement learning.

devmio: How much of the feedback provided by ChatGPT users on its answers do you think was used to train GPT-4?

Pieter Buteneers: Not necessarily GPT-4, but I believe this feedback was used to improve the feedback model and learn from specific cases. So it is only through these very specific cases that you may learn that the model is not very good at a particular task. The model can then be improved by finding new training data. OpenAI did review many of the chats and feedback and used them to improve ChatGPT. I'm not sure how big of an impact it had on GPT-4. We don't know, but there's a good chance they employ a similar model to align it.

devmio: OpenAI is aware of GPT-3 and GPT-4 hallucinations. They said that they were working on it. How does one tackle this problem?

Pieter Buteneers: One obvious way, in my opinion, is to make the model smarter, so that it knows more. More parameters and longer training come into play here: the more it knows, the less it has to make up. Another way is to use reinforcement learning to fine-tune it by telling the model, "Now you're plainly making things up, we don't like it!" As a result, the model learns that this kind of behavior is not appreciated by the person on the other end and avoids it in the future. It's the same thing they used to avoid hallucinations before, but it's remarkable how they improved on it.

devmio: GPT-4 still has the knowledge that was available on the internet before the end of 2021, or has it changed?

Pieter Buteneers: I believe it is even worse: GPT 3.5 had the knowledge that had been available on the internet before the end of December 2021 and GPT-4 only before September 2021. They most likely added some data sets later, but the majority of knowledge is from before that date, which is something Bing Chat resolves. With Bing Chat, they just search the web for new information and then combine it with the model's existing knowledge.

devmio: So, how does Bing Chat work? How are the two knowledge databases combined?

Pieter Buteneers: You can ask Chat-GPT questions regarding a text you've provided as input. With Bing Chat, all online pages scraped by the Bing search engine are used as input for the chatbot. The model can now answer based on that input.

devmio: OpenAI casually mentioned various Microsoft services that were used in the GPT-4 announcement, and it has been confirmed that GPT-4 is used in Bing Chat. What role did Microsoft play in the development of GPT-4?

Pieter Buteneers: Microsoft undoubtedly contributed financially. They invested in OpenAI because they believe it will be a highly valuable company, and they want a share in it, as well as possibly some control over it. Another thing Microsoft obviously did was provide them with computing power, allowing OpenAI to run its experiments for free or at a very low cost.

I don't know how much it cost to develop GPT-4, but I'm guessing hundreds of millions of dollars. And a large portion of that is due to the cost of computing. So getting compute for a lot less money, or even for free, is obviously a big deal. Microsoft will benefit as well because they were the first to release such a solution within Bing Chat. I'm not a huge fan of Microsoft in general, but I use Bing Chat several times every day. Therefore it's an impressive improvement in my opinion.

Many people will see Microsoft doing something that truly adds value, that is aimed at consumers rather than buyers. That is a significant shift within Microsoft, in my opinion.

But it is likely that Microsoft people partnered with OpenAI staff to figure out how to use computing power as efficiently as possible. Because, if it is given away at a low cost or virtually for free, there is an incentive to ensure that they do not use it excessively. You make sure that whatever runs on the GPUs does so as efficiently as possible. They most likely collaborated on that, at least according to some articles.

devmio: Google is integrating GPT variations into their search and into Google Bard and the result was underwhelming. What is up with Google?

Pieter Buteneers: Google is lagging behind. They weren't really prepared to launch anything, but they also stand to lose a lot more. Most people believe that Google's search engine is far superior to Bing's search engine. Even if they are familiar with Bing, most people choose Google or, if they value privacy, DuckDuckGo. Even if they are aware of it, very few people choose Bing over another search engine.

Google has a lot to lose because of its reputation and market share, while Bing has a lot to gain. It's actually far easier for Bing to debut something that isn't quite finished but generates some excitement and, at least half of the time, performs something very useful. I believe Google over-engineered its version from the start. They've been working on these kinds of technologies for a long time, but they were probably never good enough for them to launch.

Microsoft has also released a model called Copilot to help you create Microsoft Office documents. Not to be

confused with GitHub's Copilot. Google has now announced something similar because _they had to. They had to announce something comparable, but they're still a long way from being ready because they weren't in the launching mode yet.

For a long time, OpenAI has been heavily focused on GPT. I know that when ChatGPT was released, there was some confusion in Google Deepmind and throughout Google's top floors. For Google, this was a make-or-break moment. They needed to defend their territory and provide something at least comparable to what Microsoft has. I can assure you that it was an all-hands-on-deck situation at Google.

devmio: We talked about Chat-GPT before, and you said it's insane how good it is. Would you say the same about GPT-4? Is it what you expected?

Pieter Buteneers: I was expecting a minor update, and that's exactly what we received. GPT-4 is similar to ChatGPT or GPT-3.5, but slightly better, which is what I expected. When ChatGPT was released, it was different for me since it was built on a model that was already two years old at the time and it could accomplish things that I never believed conceivable in the next five to ten years. That was a huge step forward. Although ChatGPT has several faults, the transition from GPT-3 to ChatGPT is rather impressive.

Now, the transition from GPT-3 to GPT-4 is more of a step forward, demonstrating that it can do a little bit more. Perhaps, OpenAI will also offer image processing soon. If I were to guess what the next steps for GPT-5 may be, I would say it will most likely process audio and video. Maybe it will also produce images.

devmio: So GPT-5 will be a combination of DALL-E, Whisper, and others?

Pieter Buteneers: If you were to ask me, I would guess yes. GPT-5 will be more multimodal in its input and possibly even in its output, allowing it to make images.

devmio: What are your thoughts on the present state of OpenAI?

Pieter Buteneers: Not to nag too much, but I am quite frustrated with OpenAI being even more closed and claiming that they are doing so for the sake of humanity's safety. It appears to be more for the sake of their pockets than anything else.

OpenAI's blog post has some scary statements, hidden somewhere in the fine print where they claim to have very good reasons to keep it closed-source. Yet, if only a small company works on this, only a limited group of individuals will be able to identify exploits and take action. If you give it to the open-source community, everyone working in this domain around the world may become used to it and find strategies to defend themselves against misuse of the technology. The more people who understand the model, how it works, and how it operates, the more brain power you have to prevent misuse. Yet OpenAI believes it knows everything and can handle everything. It's so simple to declare it's for the greater good of humanity if it also pays your bills.

devmio: We recently spoke with Christoph Schumann from LAION. He said that open-source in AI is one of the most essential things right now and compared it to superpowers: if everyone had them, some might abuse them, but the majority would use them for good and to oppose the misuse. Do you agree?

Pieter Buteneers: I completely agree because if you have too much data or too much AI power, there are so many things you can do that you may believe are good. And you might be biased if it also fills your pockets. In practice, keeping it closed-source prevents democratic discourse on these topics from taking place.

devmio: Discourse on potential misuse, which we can't imagine right now because we don't know all of the models' capabilities, for example?

Pieter Buteneers: Indeed, and if it's closed-source, someone may discover a way to exploit the model that OpenAI hasn't. These things will bubble up on their own if you open-source it and let people play with it.

Consider DALL-E: That was one of the first truly impressive image-generation tools, but what the open-source community was able to do with stable diffusion was far more impressive. Given the advancements in AI, open-source is a positive thing. The community discovered numerous ways in which the model could be misused, as well as solutions to prevent this from happening. And now you have models that can tell whether something was formed by stable diffusion, GPT-4, or something else. If you truly open-source models, those things happen more and more on their own inside the community.

Facebook is more open-source-friendly in this aspect. Their LLaMA model, a large language model that is also quite impressive, is open-source in part. Because it is not licensed under an Apache license, you cannot use it commercially and should exercise caution. That's a bit of a pity, but at least researchers can experiment with it to a certain extent.

devmio: Thank you for your insights!

Dr. Pieter Buteneers is the former CTO of Chatlayer.ai, which was acquired by Sinch in 2020. He now leads all machine learning and AI activities at Sinch as Director of Engineering in ML&AI.

Links & References

[1] https://transfo.energy/

Interview with Muhammadreza Haghiri, AI engineer

"AI tools will push us to a world where people work smarter and not harder"

We spoke with Muhammadreza Haghiri, a developer from Iran, who is currently creating an open-source alternative to text-to-image-AI called Mann-E. He shared with us his knowledge on the technologies he uses and how he ended up utilizing them. His multi-model AI uses Stable Diffusion and Dream Booth.

by Muhammadreza Haghiri

devmio: Can you introduce yourself to our readers please? What do you do and how did you wind up there?
Muhammadreza Haghiri: I am Muhammadreza Haghiri and I was born on May 30th, 1996 in Tehran, Iran. I always had a deep love for *making stuff* and that has been developed in the field of computer science and engineering. I started programming at the age of 12 and the very first programming language I've ever learned was Visual Basic 6.0. I remember the first ever program I wrote was a calculator which could calculate the pressure a mass could apply to a surface.

I studied Computer Hardware Engineering and after graduating with a bachelor's degree, (which happened at the same time as lockdowns) I started studying AI more seriously. I had ideas, I had written code before, but I never meant to be an AI engineer before that particular time. The main motivation behind me studying AI was a TV show called Person Of Interest and that showed me the actual impacts and effects of AI on human life.

It was late 2021 and early 2022 that I found concepts of generative AI and AI Art Generators and you guessed it, I just fell in love with those AIs. So I took studies more seriously and tried to develop my own model(s) with my startup in order to generate the art forms I had in mind.

devmio: You are currently working on a text to image AI called Mann-E. Can you tell us a little bit about how it works?

Muhammadreza Haghiri: Mann-E [1](which as I explained in my blogs [2] is a play with the name Mani, which is a Persian masculine name and also name of a spiritual leader and painter in Iranian history) is a diffusion model. It means it first makes a latent space (like TV static) and develops it to an image and the image is being checked by CLIP (or other image-to-text models) in order to validate the resulting image. Then the resulting image will be shown to the user.

The main goal of Mann-E is simple, I don't want people to keep their artistic ideas/feelings inside their minds. We're currently living in a world of information floods. Our minds are flooded with a lot of information at every moment and I believe having AI tools which can help you have your thoughts as images, photos, and artworks can result in a relief.

If you would like to know how you can use Mann-E, simply pay a visit to my Github page [3]. There is a Jupyter notebook which can be imported to Google Colab. Most of the code is hidden from the end user to bring a Dall-E/Midjourney type of feelings.

devmio: On your blog you mentioned it is based on Stable Diffusion. What technologies do you use for the AI?

Muhammadreza Haghiri: This is basically what I'd call "my favorite question". I can spend hours talking about technologies I've used. First I will give you a brief history. When most of the "good" art generators weren't free, the only good base for an art generation model was VQGAN. I remember that if you mixed the results with CLIP, it could work very well. You know it was basically a game of encoder-decoder on two models.

But it wasn't satisfying enough, especially with Midjourney producing ultra-realistic images which seemed like coming straight out of an artist's mind. So what did I do? I personally thought what happens if Midjourney had an open source equivalent? And at the same time, people at Stability AI seemed to have the same thoughts. When they announced the release of Stable Diffusion it was my true call. I really liked the concept and idea and I thought it also could be useful for my project as well. This was where I spent hours on studying Stable Diffusion and related articles and research papers. It meant the world to me that finally a very good image generation model is about to be released.

We're currently living in a world of information floods. Our minds are flooded with a lot of information at every moment and I believe having AI tools which can help you have your thoughts as images, photos, and artworks can result in a relief.

Stable Diffusion was my primary technology. I loved it. I always tried to get the best I could. From prompt engineering to writing codes for getting better results. In the months following the initial release of Stable Diffusion, two great things happened. First Dream Booth for Stable Diffusion was released (and honestly, it made it much easier to fine-tune a base SD model) and also RunwayML released version 1.5 of Stable Diffusion. I had the key ingredients for making my open source Midjourney now.

To summarize: The core technology is Stable Diffusion, checkpoints are version 1.5 (by runwayml) and training/tuning process is Dream Booth. These are what made Mann-E possible. Also for programming languages, I used Python for the most part although I had to write a little bit of Ruby (mostly for web development parts of the project) and that's all that I used.

devmio: How does it differ from Dall-E, Open Journey or other models?

Muhammadreza Haghiri: That's a hard question to answer, I suppose. Especially with tons of Dream Booth or Textual Inversion models people released in the course of the last year. But if I had to tell you briefly about the differences between Dall-E and Mann-E, I'd say Mann-E isn't as expensive. To use Mann-E, you only need to fire up your Google Colab, import the notebook to your environment, and voila! You can generate an infinite amount of images in the time you are online. Compare that to Dall-E, which charges you after 50 initial generations (which I believe isn't a bad business model though).

But when it comes to SD or Open Journey, I have to say I have always been a big fan of those models and I always found something isn't right with them (specially with raw SD). It's easy for those models to quickly go off and generate completely irrelevant results to your prompt. So what could I do? I guess I try to mix good models with each other. Now I can claim Mann-E has a *multi-model architecture* under the hood which at the time isn›t easily explainable but I guess I may release an article on the topic of multi model architecture as soon as possible.

Those base models were for illustrations, paintings, concept art, analog style photographs, and double exposure photographs. With a single checkpoint and a little bit of prompt engineering you can get awesome results from the model.

devmio: You also worked on a project called Open Journey but renamed it. Can you tell us a little bit about the story here?

Muhammadreza Haghiri: Mann-E wasn't a model name first. It actually was the name of a startup company I founded back in summer 2022 and the base model was called "OpenJourney" (Open source Midjourney). Later, I found out that someone from Midjourney's team asked prompthero to change their Dream Booth model name (which I believe was midjourney-v4-style-stable-diffusion, or something similar) and they have changed it to Openjourney. I reached out to them and told them that I got the name but since I preferred to stop the company I have changed the model name to Mann-E.

And there was a happy coincidence, I believe. Someone posted the old URL to my OpenJourney project to Hacker News and my website just exploded in terms of views. I redirected that URL to the correct project and I guess it was the most unwanted recognition I got from one of my open source projects.

> *I believe AI is the future. Unlike something which just made a phase and quickly went away in 2021, Artificial Intelligence is "real" and promising.*

devmio: What do you think the future holds for AI? There is a lot of buzz around the topic, stemming from OpenAI products like ChatGPT. What is next?

Muhammadreza Haghiri: I believe AI is the future. Unlike something which just made a phase and quickly went away in 2021 (Yes, I mean Zuckerberg's Metaverse), Artificial Intelligence is "real" and promising. I see a lot of people get help from those AI tools in order to write blog posts, generate copies, generate mottos and slogans for their startups, generate thumbnails and icons, and also generate code! This means a lot, especially with tools like GPT-3 or ChatGPT it'll become more popular. On the other side we also have open source tools such as BLOOM, BLOOMZ, Flan-T5, GPT-Neo, etc. People can utilize all of these models, tools and APIs to their needs.

And I believe AI tools will push us to a world where people *work smarter and not harder*. You know you can spend 16 hours of 24 hours just to write a simple component in your react.js application, right? It's good that you love what you do but I guess it has no value to your employers and investors at the end of the day when they realize you could do that by spending 16 minutes prompting GPT-3 and doing minor edits/fixes on the resulting code. For me it's a *better world*.

Muhammadreza is a computer engineer from Iran. He has a deep love for "making" and "building" and this is what pushed him on the path of becoming a programmer. He has been an active AI engineer for the past years and is developing an open source alternative to text-to-image AI Mid Journey.

Links & References

[1] https://huggingface.co/mann-e

[2] https://haghiri75.com/en

[3] https://github.com/prp-e/mann-e

Making ML as easy as 1, 2, 3

Microsoft ML.NET: Improve Machine Learning Model Quality in Three Easy Steps

"Sometimes the shortest path is the one you know." As a .NET developer, I have been tempted many times into the world of Python by what it offers in the field of AI. I remember I was preparing a material for an event, where I was demoing .NET on Raspberry Pi. Of course, working on the Raspberry Pi, you can't resist reading the data directly from the sensors, and reading a lot of data from the sensors.

by Daniel Costea

ML.NET

Naturally, the next step came: doing something useful with that data. It's known that this the most painful part of AI. It's the lack of data. Or, as I like to say, in AI, data is not the fuel but the engine, unlike classical programming. Well, I had that data, but I didn't know what to do with it, despite its huge potential.

Ok, so I have the data and I don't know what to do with it. Extending my knowledge to another language is not necessarily a problem, but learning the related libraries is a big problem of time and integration. Let's imagine an environment where all processes are prepared for a .NET solution. Having to prepare the environment for Python can be a serious problem for a .NET developer, because it is a different ecosystem.

The moment when ML.NET was released was perfect for me because I felt like I got wings. ML.NET is easy to use for building an ML model, even for those without a data science background, and perhaps most importantly, it's a .NET framework where you only write C# code.

ML.NET is a young framework and has a lot to recover before it reaches the level of AI leaders, but I don't know if we should only see the disadvantages. Let's take a look at this ML.NET performance paper [1].

Using a 9GB Amazon review data set ML.NET trained a sentiment-analysis model with 95% accuracy. Other popular machine learning frameworks failed to process the dataset due to memory errors. Training on 10% of the data set, to let all the frameworks complete training, ML.NET demonstrated the highest speed and accuracy. The performance evaluation found similar results in other machine learning scenarios, including click-through rate prediction and flight delay prediction (**figure 1**).

Model Builder (AutoML)

With the Model Builder, ML.NET provides developers with a very useful tool for training ML models for a predefined time interval, starting from a data set and ending with the selection of the best trainer for the selected scenario. The selection is made according to the quality of the explored models. Therefore, the obtained ML model can be consumed immediately. In addition, Model Builder is able to generate boilerplate code with all steps taken interactively, providing a good starting point in the world of AI (**figure 2**).

Model Builder is an extraordinary tool that allows you to preselect the features that you want to include

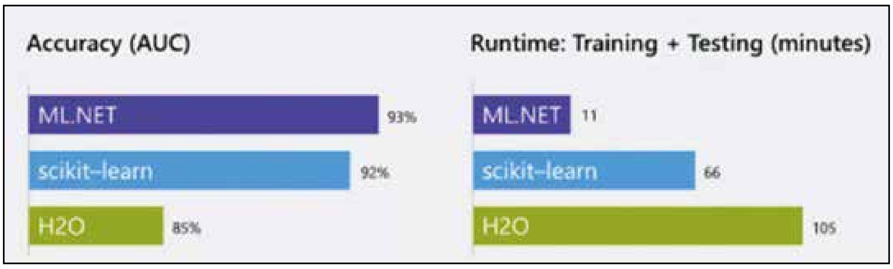


Fig. 1: Machine learning frameworks comparison

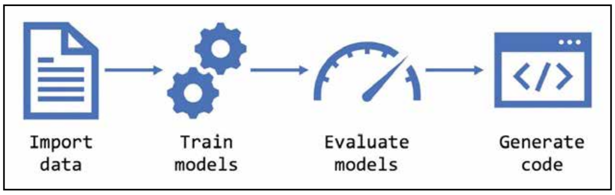


Fig. 2: Four steps with the model generator

in building the model. However, it doesn't give you any suggestions about which features are more relevant. On the other hand, a model that has too many features will require more time for training and for prediction. Many times, some features alter the quality of the built model more than they help. This means that by carefully reducing the dimensionality of the ML model, we could increase the accuracy and performance of the model. Let's keep this in mind, because we'll deal with it a little further down in this article.

I was saying earlier that Model Builder is capable of generating boilerplate code. But from a software development perspective, we'd like to automate this training process of the ML model and we aren't satisfied with the code generated.

But I have good news. At its core, Model Builder has the AutoML, which provides the entire ML model training experience. It would be great if we could take control of this code and take it a few steps further by analyzing the model's quality, in order to make some improvements.

Assuming we loaded some data in trainingDataView object, these are the few lines that trigger the AutoML experiencing for multi-class classification (this is the scenario I chose for this article) in a given time interval (listing 1).

And here the AutoML Output (listing 2).

Listing 1

```
Context = new MLContext(seed: 1);
var features = new string[] { "Temperature", "Temperature2",
                    "Luminosity", "Infrared", "Distance", "PIR", "Humidity"};
var experimentPipeline = Context.Transforms.Conversion.
MapValueToKey(Label)
   .Append(Context.Transforms.Concatenate(Features, features))
   .Append(Context.Auto().MultiClassification());

var experimentSettings = new AutoMLExperimentSettings
{
   Pipeline = experimentPipeline,
   MaxExperimentTimeInSeconds = time
};

var experiment = Context.Auto()
   .CreateExperiment(experimentSettings)
   .SetDataset(trainingDataView, testingDataView)
   .SetEvaluateMetric(MulticlassClassificationMetric.MicroAccuracy,
                                                   Label, PredictedLabel);

var experimentResult = await experiment.RunAsync();
```

Listing 2

```
Best trainer: FastTreeOva        Accuracy: 0.926   Training time: 338
-------------------------------------------------------------------------------
   MicroAccuracy     MacroAccuracy        LogLoss      LogLossReduction
   0.926             0.929                0.235        0.826
```

Model Analysis

"Measure twice and cut once"

I would like to go a little further and see how we can improve an ML model. Again, we're doing this without having any knowledge of Data Science, but don't get me wrong, having a basic understanding of Data Science would help you better understand what you're doing.

We might think that the most difficult part is building the training model, because, obviously, that's where the magic of things seems to happen, but for this part, Model Builder helps us with great success using its AutoML. In fact, the most difficult part is to build a good enough model, and for this we will rely on the correlation matrix analysis and the PFI analysis. In fact, if we could automate these steps, it would be extraordinary, because for each model obtained through AutoML we could go further obtaining a better model with minimal effort.

Correlation Matrix

In machine learning, a Correlation Matrix (also known as a heatmap) is a table that displays the correlation coefficients between all possible pairs of the existing features (like the Cartesian product).

Generally speaking, having more features is a good thing, but this increases the dimensionality of our model. Therefore, it increases the model performance (training time and prediction time, as well). But do we really need all these features? Some might redundant.

A value closer to 0 means low (or not) correlated features. A value closer to 1 means high correlated features, and a value closer to -1 means inverted high correlated features.

Listing 3

```
Correlation Matrix, threshold: 0.9
-------------------------------------------------------------------------------------------
             Temperatu  Temperatu Luminosit Infrared Distance  PIR      Humidity
 Temperature  1.0000    0.9998    0.4831    0.5865   -0.2873  -0.0020  0.0607
 Temperature2 0.9998    1.0000    0.4822    0.5855   -0.2859  -0.0018  0.0621
  Luminosity  0.4831    0.4822    1.0000    0.4388   -0.5428   0.1175  0.0457
    Infrared  0.5865    0.5855    0.4388    1.0000   -0.3765   0.0359  0.0051
    Distance -0.2873   -0.2859   -0.5428   -0.3765    1.0000  -0.1412 -0.0824
         PIR -0.0020   -0.0018    0.1175    0.0359   -0.1412   1.0000  0.0809
    Humidity  0.0607    0.0621    0.0457    0.0051   -0.0824   0.0809  1.0000
-------------------------------------------------------------------------------------------
No Feature       vs. Feature        Rate
1. Temperature   vs. Temperature2   0.9998
```

```
var matrix = Correlation.PearsonMatrix(dataArray);
```

Correlation Matrix output (using a predefined threshold of 0,9) (listing 3).

Since Temperature2 and Temperature seem to be redundant (maybe in practice the distance between the sensors is small!), we can consider removing one of them.

Permutation Feature Importance

Using Permutation Feature Importance (PFI), we learn how to interpret ML.NET machine learning model predictions, because PFI shows the relative contribution each feature makes to a prediction. The way PFI works is by randomly shuffling data one feature at a time for the entire dataset, calculating how much the performance metric of interest decreases. The larger the change, the more important that feature is. Additionally, by highlighting the most relevant features, we can focus on using a subset of more meaningful features which can potentially reduce noise and training time. We have to decide carefully which features we don't need, because by removing some not-so-relevant features we risk introducing bias in our model.

You can see occasionally negative values in PFI results. In those cases, the predictions on the shuffled (or noisy) data happened to be more accurate than the real data. This happens when the feature didn't matter (should have had an importance close to 0), but random chance caused the predictions on shuffled data to be more accurate. This is more common with small datasets, like the one in this example, because there is more room for luck or chance.

The code may look a little more complicated because we need to replicate the pipeline used for AutoML (listing 4).

PFI metrics output (using a predefined threshold of 0.01) (listing 5)

We extract the best performing trainer from the *experimentResult* object obtained by running AutoML, and proceed to its evaluation.

```
var predictions = experimentResult.BestRun.Model.
                                        Transform(testingData);
var metrics = Context.MulticlassClassification.Evaluate(predictions);
```

Listing 4

```
var model = experimentResult.Model as Microsoft.ML.Data.
TransformerChain<Microsoft.ML.ITransformer>;
var transformedTrainingData = experimentResult.Model.
Transform(trainingData);

var pfi = Context.MulticlassClassification
.PermutationFeatureImportance(model.LastTransformer,
transformedTrainingData, permutationCount: 3);

var metrics = pfi.Select(p => (p.Key, p.Value.MicroAccuracy)).OrderBy(m
=> m.MicroAccuracy.Mean);
```

Listing 5

```
PFI (by MicroAccuracy), threshold: 0.05
-------------------------------------------------------------------------
  No Feature       MicroAccuracy   95% Mean
  1. Infrared      -0.2467         0.0274
  2. Luminosity    -0.2181         0.0121
  3. Temperature   -0.1224         0.0019
  4. Distance      -0.0795         0.0025
  5. Temperature2  -0.0257         0.0043 (candidate for deletion!)
  6. CreatedAt     -0.0186         0.0074 (candidate for deletion!)
  7. PIR           -0.0076         0.0033 (candidate for deletion!)
  8. Humidity       0.0000         0.0000 (candidate for deletion!)
```

Evaluation results with the original dataset (listing 6).

Let's get rid of the redundant features (indicated by the Correlation Matrix) and the irrelevant features (indicated by the PFI) from dataset and proceed to another experimentation with the diminished dataset. Let's check the evaluation settings again.

Now the diminished list of the features look like this:

```
var features = { "Temperature", "Luminosity", "Infrared", "Distance" };
```

Evaluation results with diminished (without redundant or irrelevant) set of features (listing 7).

Conclusion

"Less is more"

If we have a dataset with many unprocessed features, the PFI will mark the candidate features for deletion. By removing one or more of these features and retraining the model we may get a better model (AutoML will take care of finding the best trainer for a given set of features).

If the Correlation Matrix identifies highly correlated features, we can delete these features one by one, perform the retraining and check if the model gets better with the new set of features.

One nice improvement of the source code associated with this article would be to automate the deletion of the most irrelevant features (using PFI) or the most correlated (redundant) features (using Correlation Matrix).

You can find the code associated with this article here [2] and more articles about ML.NET on my blog [3].

Listing 6

```
Best trainer: FastTreeOva          Accuracy: 0.926  Training time: 338
----------------------------------------------------------------------
   MicroAccuracy      MacroAccuracy     LogLoss   LogLossReduction
       0.926              0.929          0.235         0.826
```

Listing 7

```
Best trainer: FastTreeOva          Accuracy: 0.943  Training time: 342
----------------------------------------------------------------------
 MicroAccuracy     MacroAccuracy      LogLoss    LogLossReduction
     0.943             0.942           0.207          0.847
```

Daniel Costea is a trainer, developer, speaker and Microsoft MVP. In the last four years, he has delivered more than 1200 hours of training on .NET web technologies to more than 600 students. In the last two years, he has been a speaker on .NET technologies at more than 20 IT conferences in Romania, Europe and the USA. Excited by their great potential, Daniel is expanding the learning focus to new trends such as the internet of things and machine learning. Daniel strongly believes in the power of community and is a co-organizer of ApexVox, a free .NET conference for developers in Romania.

Links & References

[1] https://arxiv.org/pdf/1905.05715.pdf

[2] https://github.com/dcostea/AutoMLSample

[3] http://apexcode.ro/

Imprint

Publisher
Software & Support Media GmbH

Editorial Office Address
Software & Support Media
Schwedlerstraße 8
60314 Frankfurt, Germany

Tel: +49 (0) 69 630089-0
Fax: +49 (0) 69 630089-69
Web: www.sandsmedia.com
Layout: meat* – concept and design

The views expressed are solely those of the authors and do not reflect the views or position of their firm, any of their clients, or Publisher. Regarding the information, Publisher disclaims all warranties as to the accuracy, completeness, or adequacy of any information, and is not responsible for any errors, omissions, inadequacies, misuse, or the consequences of using any information provided by Publisher.