Acumos

An Open Source AI Machine Learning Platform

Acumos AI

ACUMOS

An Open Source AI Machine Learning Platform

By AT&T and The Linux Foundation

Copyright 2018 AT&T Intellectual Property. All rights reserved.

Licensed under the CC by 4.0 license.

1 Executive Summary

Over the last several years, there has been tremendous excitement around and interest in the potential for machine learning (ML) technologies applied to Artificial Intelligence (AI). Over $11 billion has been invested by venture capitalists in new AI startups during 2014-16, with an estimated $6 billion more in 2017. Tech giants spent between $20-30 billion on AI in 2016 with technology and finance leading the way. Such investment have enabled numerous consumer services such as Alexa and Siri to become mainstream. Despite those successes, development and deployment of AI applications remain expensive in terms of time and the need for specialized talents. The lack of any common frameworks for development have kept AI applications from becoming mainstream as a general solution to IT problems.

Acumos is an Open Source Platform, which supports training, integration and deployment of AI models. By creating an open source community around Machine Learning, AT&T will accelerate the transition to AI-based software across a wide range of industrial and commercial problems and reach a critical mass of applications. Acumos drives a data-centric process for creating machine learning applications.

Acumos empowers data scientists to publish adaptive AI models, while shielding them from the need to custom develop fully integrated solutions. It packages each model into an independent, containerized microservice, which is fully interoperable with any other Acumos microservice, regardless of whether it was built with TensorFlow, SciKit Learn, RCloud, H2O or any other supported toolkit. Models built with any of these tools or any supported language, including Java, Python and R, can be automatically onboarded, packaged and cataloged. These microservices are easy to integrate into practical applications, for any software developer, even without a background in data science or knowledge of any specialized AI development tools.

Acumos provides a **Marketplace** mechanism for data-powered decision making and artificial intelligence solutions. Self-organized peer groups

across one company or across multiple domains can securely share information among themselves on how AI solutions perform, with ratings, popularity statistics and user-provided reviews, to apply crowd sourcing to software development. The platform aids communication between data scientists and application developers in order to automate the process of user feedback for selecting models while automating error reporting and software updates for models that have been acquired and deployed through the marketplace.

Software developers will transition from a code-writing and editing toolkit into a classroom-like code training process. AI models can be acquired from the marketplace, trained, graded on their ability to analyze datasets, and integrated, automatically, into completed solutions. They can access encapsulated AI models, without knowing the details of how they work, and connect them to a variety of data sources, using a range of data adaptation brokers, to build complex applications through a simple chaining process.

Unlike many proprietary systems, Acumos is not tied to any one run-time infrastructure or cloud service. Acumos creates an extensible mechanism for packaging, sharing, licensing and deploying AI models and publishes them in a secure catalog which is easily distributed between peer systems. Catalogs contain detailed information on ownership and execution requirements of individual microservices so that these can be easily searched with the Acumos Portal to make model selection and deployment a simple process.

Acumos includes a graphical tool, called **Design Studio**, for chaining together multiple models, data translation tools, filters and output adapters into a full end-to-end solution that can be deployed into any run-time environment, such as a cloud service, including AIC, Azure and AWS, a private datacenter, or a specialized hardware environment designed to accelerate AI applications. Acumos only requires a container management facility, like Docker, to deploy and execute portable general purpose applications.

2 Introduction

In the past several years, great strides have been made in Artificial Intelligence software. The introduction of new software designs and revolutionary new hardware infrastructures have begun to show the amazing potential of AI technology to change the way that we interact with and use computers. AI has had a tremendous amount of coverage in the popular press in recent years and "machine learning" seems to be the trending term in software these days, but how much headway has artificial intelligence made in actual market penetration?

According to a June 2017 McKinsey and Company report entitled "Artificial Intelligence The Next Digital Frontier?" (McKinsey Global Institute, 2017) gains in AI investment have been limited in scope and not related to a wide segment of commercial firms. In fact, during 2016, companies invested between $26B and $39B in this field, which is a substantial amount and represents a high rate of growth, about triple the amount invested in 2013. But $20 to $30 billion of that investment has been made by a few giant tech firms, dominated by Google and Baidu. McKinsey's survey indicates that, of those firms who are aware of AI technology and its potential, only about 20% have actually begun to adopt it in their own businesses. Another 41% of firms said they are still uncertain about the benefits that AI had to offer in their industries and are only investigating or experimenting with potential opportunities while the other 39% of industry is still just "contemplating" the technology. Looking at this from the perspective of the venture capital market, only about 2%-3% of VC funding went into AI in 2016, as compared to 60% going to information technology, in general.

The real breakthrough in AI in recent years has been the shift to machine learning as the preferred technology behind the most successful efforts. But the focus of the few industry leaders who are deploying the technology remains in areas which were pioneered many years ago, such as machine vision and natural language processing. These fields relate directly to the areas where those tech giants have put their primary efforts such as voice-driven digital assistants, robotics and self-driving cars.

What is happening is a growing gap between the leaders in AI and the bulk of industry, which threatens to leave much of the commercial base behind. As more tools are built to satisfy the requirements of a few narrow areas of inquiry by the dominant companies in the field, other companies and the problems that they would like to tackle are neglected. As technology becomes more specialized, a unified framework for general purpose AI development has, so far, failed to emerge. Data scientists have become more engaged in areas of early success, but proportionately less effort is being spent on some of the very valuable industrial innovations that many companies would like to see addressed.

What is needed is a common framework around which a larger ecosystem can develop. Practical solutions will never be built with just one technology. Instead, AI must be employed in tandem with other, less specialized, solutions to tackle the real world problems of the majority of industrial and commercial firms.

3 A New Model For Software Development

Machine Learning has created a new paradigm for software development which promises to substantially change the nature of computer software in the near future. For a long time, AI was dominated by expert systems that were only partly data-driven. But, as the ability to store and process very large amounts of data has grown over recent years, the field has moved to more data-centric technologies. While data driven analytic methods have been around for many decades, they were originally quite limited in scope and application. Some techniques, such a regression analysis, are applicable only to quantitative problems making use of extrapolation from known quantities. Other approaches require well defined, randomized and carefully analyzed datasets.

In problem domains in which the data is very complex, hard to structure or organize and which is not easily addressed through analytic tools alone, the increase in speed and storage capacity of modern systems has made it possible to approach problems in a highly data-centric way. Today, we are seeing the emergence of generalized algorithms which draw from large datasets, including, not just supervised learning from well-organized and labeled data, but also unsupervised techniques based upon unstructured input, such as reinforcement learning. Not all problems fall into these categories, but for those that do, a new model of software development is needed.

Reinforcement learning systems are trained from their own experience, in principle allowing them to operate in domains where human expertise is lacking. There has been rapid progress towards this goal, using deep neural networks trained by reinforcement learning, in particular, for problems which were not successfully solved through analytic methods. For example, systems for playing games like Go and Chess, have proven highly successful, where systems like **AlphaGo Zero** (Silver, 2017) have exceeded, not only human players, but even AI programs developed by humans following proven strategies. Reinforcement learning is based solely upon empirical results from successive contests in which the program plays against itself without any supervision and without augmentation from human-designed analytics. The key ingredient for

these new approaches is the ability to acquire a vast quantity of data in a very short period of time.

The old model of software development – code, test, debug, retest, deploy – is not a good methodology for building machine learning software of this type, because it leaves out the training process. Training of the software must become a full-fledged part of the development cycle. Code must first be developed and then trained. This training often requires its own tools including a functional user interface, data acquisition code and storage management which makes it possible for the program to acquire the data that it needs. And, since the code must be built and then trained before it can be fully debugged and since the training code, itself, must also be built and debugged before any training can commence, the overall development process becomes much more complex.

This new advanced learning requirement changes the focus of software development away from conventional coding tools toward a new set of training-oriented tools which will dominate the technology of the future. Data scientists will continue to program the models which form the basis for AI solutions, but much of the real work will shift to training the models with ever-larger quantities of data. It is this training and the data it uses that will differentiate one solution from another, rather than the algorithm employed by the solution. That is to say, a well-trained program with an inferior algorithm may outperform a poorly trained one, even if the algorithm itself is better. In fact, future developers will “shop around” for a set of likely algorithms and then present them all with extensive data representing the actual problem to be solved. The selection of a successful program will then be done by testing each one on its ability to recognize conditions in a test dataset. In other words, program candidates will be trained and then presented with a final exam to test their effectiveness before a program will be selected from the candidate pool.

This change will have two significant impacts on the tools and methods being used in software development. First of all, data scientists will have less to do with the training of their programs than developers traditionally have had with debugging their code. In conventional software coding, the developer understands the task to be programmed in the minutest detail. Training a machine learning algorithm, on the other hand, is a tedious and long-running process that does not really require the skillset of a data scientist. The need is more for a skilled practitioner of the subject matter who can help to guide the machine learning component to the right conclusions. It is obvious that the training of an autonomous vehicle should be left to a skilled driver, rather than to a software developer. It is far better to let data scientists work on learning algorithms than to waste their time on organizing training sessions in a field that they know little about. The tools that people use to train machine learning algorithms will have to be ones that are tuned to the skillset of the trainer, rather than traditional programming tools with which data scientists are most familiar. Knowledge of data storage tools and schema, or expertise in modeling languages and mathematics should not be a prerequisite for using these training tools. In fact, training of the algorithm will, itself, become a more and more sophisticated application with its own user interface and data sources.

Secondly, the machine learning aspects of any skill are not directly related to the actual operation of a device. Models may learn to do things, but they cannot actually do those things without additional programming. Therefore, we will see a separation between the AI algorithm, which is designed by machine learning experts and trained via a process of feedback from real-world results, and the program which carries out the task for which that intelligence is being employed. As an example, a facial recognition algorithm may be trained to identify a set or class of faces. But the application of that facial recognition requires that recognizer to be combined with some actual application, such as a search algorithm which captures images from a series of surveillance cameras in order to locate a

single face, or a security system which is able to sound alarms or activate door locks. The application which accesses the set of cameras looking for a match, in the former case, or opens and closes a door on recognizing an authorized face, in the latter case, has little to do with AI or machine learning. Thus, the specialized programming tools used by data scientists are poorly suited even to the needs of a software developer who writes applications which make use of some AI components.

Machine learning programs will have to be deployed in a componentized form that can easily be incorporated into a specific solution, similar to the way that libraries of mathematical functions are marketed as separate components. But, utilizing AI in a general program is not the same as selecting a complex signal processing algorithm from a math library and invoking it. A math routine will always return the same results and two implementations of the same routine may vary in performance, but never in the quality of the result. The differences between AI algorithms, on the other hand, can be quite profound. One may do better at men's faces than at women's. Another may have fewer false positives but more false negatives. The test of an AI algorithm will depend upon how well it does on its "final exam" organized to represent the actual conditions of the application which uses the algorithm being tested. The development tools will need to make it possible to defer selection of a machine learning algorithm for an application until the data has been collected and the training is complete. This is a big departure from the status quo in software construction.

4 Applying AI to Commercial and Industrial Problems

Besides allowing the general IT developer to make use of machine learning algorithms without the need for a data science team to be engaged in the development process, the Acumos community also seeks to broaden the application of AI and machine learning beyond the most obvious, and most academically interesting, use cases.

Commercial and industrial concerns are by-and-large more interested in solving problems which involve the majority of human decision making in their own enterprises in order to improve productivity, improve performance and reduce the cost of production or service delivery. Areas of interest like autonomous vehicles are exciting and get a lot of press attention today, but they do not deal with the main cost components for most enterprises, especially for companies engaged in communications or data center operations or customer services of various kinds. By coming together in an open source community, which is not driven by academic interests alone, but also by business requirements, we can begin to attack those business problems which contribute most to our costs but do not, in themselves, add new products to the market. This is where a real opportunity exists to change the way the world does business.

There are a number of areas that we have identified which are of immediate interest to a broad set of commercial and industrial enterprises. The following are just a few examples to illustrate the potential of applying machine learning to problems which are, perhaps, less dramatic than machine vision, but extremely important to the balance sheets of many companies.

NETWORKING

Companies engaged in communications, data traffic, media streaming

and other areas must contend with anomalies in the flow of traffic across our systems and networks. These anomalies may occur during periods of peak demand or when demand shifts from one geographical area to another or when it shifts from one application to another. Machine Learning algorithms can be used to identify the normal flow of traffic through a network and identify whenever that normal pattern changes. By responding quickly to such changes, a network can adapt quickly, as necessary, so as to minimize the impact of a change on customer experience. By automating the spinning up of additional media servers, or launching servers in a new location where demand has suddenly spiked, the network can adjust to these impacts in ways that human controlled processes, which must operate at human speeds and on a human schedule, cannot easily accommodate.

Similarly, when failures in network links or sudden loss of servers occur due to either hardware or software failures, a traffic anomaly will follow. An automated response based upon the detection of such a networking anomaly can provide rapid replacement or rerouting of service to correct the failure. Not only will this improve customer experience and satisfaction but it also can reduce the need for redundant equipment. If the response of the network can be tailored to the type of anomaly detected, in real time, a small quantity of redundant equipment can be sufficient to correct a wide variety of potential system failures. Without such an automated response, the network must be provisioned to handle any and all failures simultaneously since the network designers have no way of knowing which failure will occur in the future. Fully redundant systems are costly to build and also add to the cost of operation. Self-healing responses based upon the detection of actual failures in real time can make use of a small complement of spare equipment to adapt only to the problem which is detected and only until the root cause of the problem is corrected. Thus, a small amount of spare capacity can be used to address a wide variety of potential failures since it only has to deal with one or two failures at any one time.

The result is a nearly seamless recovery from any failure, but with a significant reduction in unneeded investments in redundant systems deployed in case of failure.

INFRASTRUCTURE

Data center infrastructure must be in place before a service can make use of it. Today, the purchase and deployment of infrastructure, whether for manufacturing, IT operations or for online services, is done based upon estimated future needs, which usually relies upon sales estimates or usage predictions. Such estimates must make use of worst case assumptions and planning for peak load, but when the estimates for multiple applications or services are combined, any errors are magnified.

- Even though peak demand for a billing process may occur at a different time of the day or month than peak demand for streaming media, it is very difficult to combine the individual estimates and adjust the hardware needs accordingly.
- Furthermore, errors accumulate so that if one estimate of equipment needs is too high in one quarter, it is rarely adjusted down in the next quarter, so over time each wrong guess for each component adds up to a lot of overcapacity, which may never be detected because the overhead is baked into the design. All extra capacity is accounted for in business plans and, as such, assumed to be necessary, even when that is not the case.
- And, finally, human assessments of this kind are often unreliable because they may be driven by the need to justify the importance of an investment as much as by the need to minimize capital expenditures.

A machine learning algorithm, on the other hand, can be trained on actual aggregate load to predict the rate of growth, directly. A true measure of changes in load over time, including sudden load bursts at peak times, better predict hardware requirements thereby minimizing investment.

Other infrastructure investments are related to decisions about the proper time to repair hardware failures in the field. It is well-known that continuous repair of equipment each time any failure occurs, is costly and inefficient. By deploying some redundancy, it is possible to wait until the amount of failed equipment reaches a critical level before going into a data center and replacing a number of servers at once. This strategy cuts down on operations costs, since repairs are not carried out as frequently, and in service disruptions, since repairs can be done at the most convenient times, such as when load is at a minimum. But, beyond a certain point, leaving failed equipment unrepaired can risk that an occasional peak load will result in an unexpected service disruption. So, in the absence of real data on actual infrastructure needs and an accurate assessment of possible vulnerabilities, most operations will still play it safe and over-estimate hardware sparing needs.

The decision on when to replace infrastructure, minimizing cost without adding to the risk of catastrophic failures is a complex one which can benefit from advanced metrics and adaptive analysis. Too often, companies attempt to use simplified policies based upon fixed failure thresholds or monthly work schedules which can prove highly inaccurate. Applying machine learning and AI to this process can increase the efficiency of setting maintenance schedules while reducing the inventory of replacement parts by driving these processes from conditions in the field, load and even market conditions, instead of blind scheduling.

SECURITY

Growing threats to enterprises of all kinds from hackers, competitors and

foreign governments to break into computer systems for the purpose of stealing data, monitoring customers, or destroying equipment have become critical. Conventional security analytics must be programmed to detect an attack by anticipated threats using specific threat detection software built to deal with known attacks. Machine learning can be employed as an effective countermeasure to many such threats, including both known and unknown attacks. AI software can learn new patterns of attack, as they occur, by detecting sudden changes in data traffic and comparing them to earlier experiences in order to select and deploy countermeasures. Not only could such software adapt to entirely new threats, as they are launched, but the software can also become better at detecting security threats as it learns from each successive threat. Machine learning based analytic software could be installed once and continue to improve its responses over time based upon experience, so companies would not be required to continuously install updates and patches in order to keep ahead of the latest threats.

Machine learning can also be applied to the task of actively scanning software, systems and networks for vulnerabilities. Weaknesses in design of software and networks could be detected and repaired even before they are exploited. This scanning could include searching for insecure operating system versions, or detecting packets from unsecured sources entering restricted areas of a network or probing for ports and addresses that should be blocked, but are not. The specific vulnerabilities are constantly changing, as is the software which is deployed, so it is very difficult to programmatically set security policies and apply them. An automated security scan with the ability to learn could continuously update the scanning parameters to insure that any new vulnerabilities introduced by changes in connectivity or software version are automatically detected and corrected.

VISUAL INSPECTION

Image recognition software is another area where industrial applications will rapidly spring up. ML algorithms can be trained to recognize components used in various assemblies so that they can be linked to various quality control applications. A feature recognition module, for example, can be trained, in one instance, to recognize the parts of an aircraft and a robot can automatically move from surface to surface looking for cracks in wings, engines and fuselage, imperfectly sealed compartments, loose hoses, and many other potential safety issues, automating the inspection process so that hundreds of robotic inspectors can operate simultaneously to reduce the turnaround time of an aircraft. On an assembly line, the same feature recognition software can be trained to identify each piece of equipment on the line and look for visible indicators of damage or failure on manufacturing equipment, thereby proactively alerting the plant to problems before faulty product is produced. The same sort of visual inspection software could be applied to automobile maintenance to ensure that physical damage is automatically detected and addressed in the same way that electronic metrics are monitored.

Automated visual recognition models can be especially useful in the operation of drones which rely on this technology to locate flight waypoints, and position themselves precisely around inspection targets. Such programs used in conjunction with drones can automate the broadcasting of sporting events allowing different drone cameras to follow preselected targets in comprehensive ways.

CUSTOMER CARE

Machine learning will also transform customer service, technical support and other customer care applications. Chatbots are already making use

of flagged words to automate customer support interactions by taking customers through a scripted interaction based upon keyword matching. This technology can be extended to support topic models which classify natural language queries against a learned set of topic categories. By aggregating and scoring customer responses to an interaction with human agents, a machine learning algorithm can automatically "answer" a customer's question classifying the customer request to a known topic and playing back a human's response, recorded earlier, to a similar question. Only when the question is not understood, or when the answer does not satisfy the customer, would a human agent be brought into the loop. In this way, the algorithm can accumulate an unlimited number of topics and responses associated with those topics over time, gradually reducing the need to refer inquiries to the human agent. The effect will be to provide a customer care solution which will scale along with demand. The benefit to the customer is that response time for any inquiry will be the same, regardless of how many customers are calling into the system. Without a human in the loop, there is no need to queue up requests to a limited number of agents. AI bots can be spun up to handle any number of customer calls simultaneously. What is key to this experience is that human agents can act as trainers to the AI models by handling all the exceptional cases, such as new inquiries or customers who do not get satisfactory service from a bot. For the customer, this will result in a best-of-both-worlds experience: rapid responses for well understood problems and automatic transfer to a live agent for unusual situations.

Because this technology is based upon the classification of a "topic" regardless of the format of the input, the system can be adapted to handle chat, web, voice and video interactions with a single model.

5 The Purpose and Mission of the Acumos Platform

Acumos employs the open source collaboration model to achieve flexibility and create a highly adaptable industry-wide framework for building, training, integrating and deploying machine learning solutions. In order to simplify the development process and make it understandable to a wide audience of developers, Acumos breaks the development flow into four distinct steps. First, models are onboarded to an Acumos-compliant platform and packaged as distinct microservices with a component blueprint describing the microservice API and dependencies. Second, the model is packaged into a training application which can be deployed to an appropriate training environment from which data can be acquired and cached for later retraining, if needed. Third, a reference to the trained model, which is called a ***Predictor***, is published into a catalog which can be shared across a community where other developers can find it, discuss it, review it, and create a full solution by using the Predictor and "chaining" to other components which employ the Predictor to make decisions but provide many of the more conventional capabilities that are required to act on them. Finally, the entire solution is packed into a ***Docker container*** which can be deployed to an appropriate runtime environment where it can be executed. While some of the components in this packaged solution are used to access data and implement functionality, others can be used to transfer new data records to the Acumos platform where they can be added to the cached datasets and used for additional training in a continuous learning process. Figure 1 is a diagram of these four distinct stages of the development and deployment process.

The reasons for approaching the problem in this way are to separate the very specialized functions of model design and data management from the complexities of service development and application lifecycle. By keeping these aspects of AI development distinct, we can make the process quicker, more reliable and open to a much larger community of users.

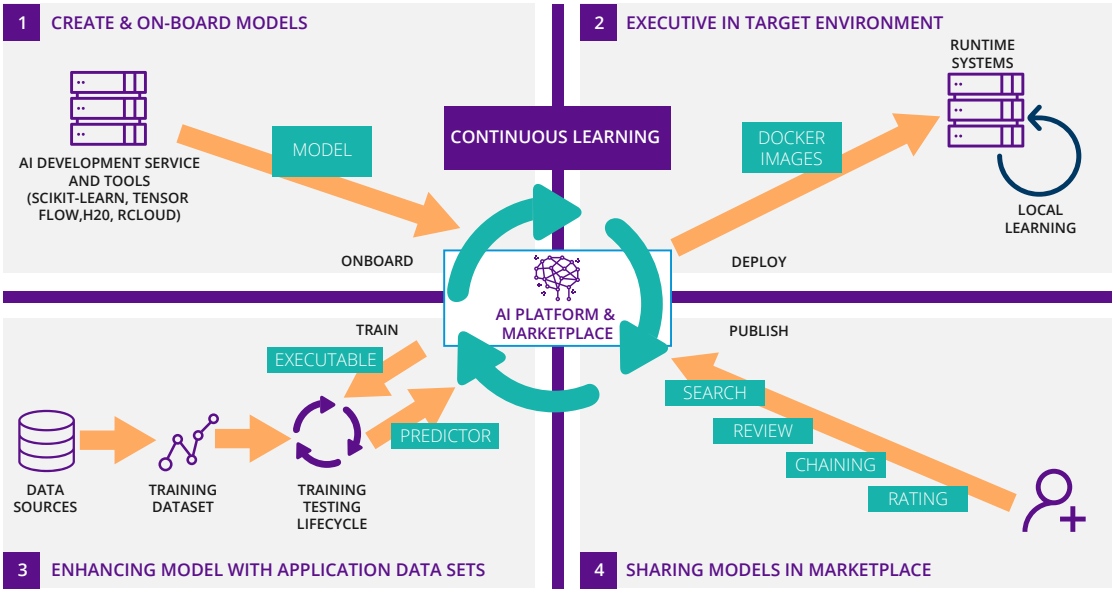

Figure 1. Four stages of AI development supported by Acumos

STEP 1 - CREATING AND ONBOARDING NEW AI MODELS

As illustrated in the upper left quadrant of Figure 1, Acumos does not include a specialized platform for developing AI solutions aimed at data scientists. There are already a large number of excellent development tools for building neural nets, classifiers, clustering algorithms and other types of AI components. But these tools do not make it easy to integrate with other components. Either the tools are tied to a particular execution platform, such as a specific cloud service, or they are very specialized to the needs of the data scientist and difficult to use for the average software developer. Each tool is implemented around some language and a specific set of compatible libraries. All these things narrow the audience for any tool and limit access to previous work by requiring compatible components to adhere to some standard.

Acumos is a way to harmonize solutions across the full range of existing and future AI tools and technologies. Initially, Acumos supports such toolkits as SciKit Learn, TensorFlow, H2O, RCloud and various programing languages. A portable Acumos library packages the products of all

these tools in such a way that they are all interoperable. By leveraging an open framework, for which all of the source code is readily available and adaptable, the expectation is that the number of compatible SDK's and languages will grow over time, as the Acumos community grows. By wrapping programs in a container, making each toolkit and language interoperable with the others, the range of available, compatible solutions that Acumos can use is large and will only grow over time.

For the foreseeable future, it is likely that no single, all-purpose AI hardware platform or development kit will dominate software design for machine learning solutions. Each problem and each approach is likely to require a different set of tools. Therefore, a platform which harmonizes many solutions by packaging them into interoperable microservices will be the best way to make sure that solutions will be useful for any target development community. Furthermore, since the field is changing rapidly, the future will include a wide array of tools which do not currently exist and probably have not yet been conceived. So any attempt to "standardize" on one approach or toolkit at this stage is doomed to failure. Instead, a tool which focuses on interoperability will help to keep today's solutions relevant for future applications.

STEP 2 - TRAINING THE BASIC AI MODEL

What is common across different AI platforms and tools is this new model of software development in which code must be enhanced through training. Machine Learning is an enabler to a wide variety of AI technologies and about 60% of the investment in 2016 for AI has gone to the development of machine learning (McKinsey Global Institute, 2017). The application of machine learning to business problems depends, first of all, on the ability to train software by providing specialized data sets that represent a variety of conditions for the model to identify and act upon. So the ability to quickly and reliably train software on any dataset is a component of every machine learning problem.

Since all machine learning is data-driven, algorithms based on machine learning technologies are substantially reusable in ways that prior generations of software were not. Conventional software tools are designed to address the problems of editing, inspecting and analyzing code to simplify debugging and reprogramming.

In machine learning problems, the process is more about taking a good basic algorithm and applying it to a new dataset in order to solve a new problem without developing any code. But in order to train the algorithm, it must be executed in an environment in which the training data is available. The ability to securely exchange, retrain and license a general purpose solution is a key part of a commercial machine learning platform. While entirely academic problems can be trained in a relatively isolated development environment to which the data is securely imported, real world training requires moving the model to an environment where data can be conveniently and rapidly acquired. In other cases, such as robotics, crowd sourced datasets or mobile applications, for example, the data may only be available in some very specific runtime environment.

Acumos takes these packaged models and moves them from a secure development machine to a secure runtime and exposes the model to training data without the need for a developer to change the model in any way. This is done using custom training clients, data access and data caching tools, making it easy to assemble a specialized training application for each machine learning model. Acumos provides the training and testing interface needed to turn a basic model into a Predictor which has been trained to perform a specific function.

STEP 3 - PUBLISHING MODELS TO A CATALOG

Data-driven machine learning models can be trained to recognize a pattern and to classify patterns reliably, but they cannot learn to do different things, once trained. Therefore, the third goal of a common AI development platform is making it easy to connect AI algorithms with

different adapters that can apply the knowledge acquired in a training process to specific applications. For instance, if you are building a corporate access control system, it may be a simple matter to train a machine learning algorithm to recognize an employee of a company from personnel records, but before that model can be used to open and close doors, some kind of an adapter is required to actuate a lock when a known employee approaches the building entrance.

In order to build a working system, it is necessary that developers locate components which implement specific functions, evaluate what those components do and how they interface to external systems and data sources. Acumos does this by creating catalogs of useful functionality that can be searched and explored, reviewed and rated. Once a model is identified in a catalog, it can be acquired from the original developer and employed as a component in a complete solution. It is the catalog which connects a component developer with a population of potential users and facilitates that acquisition, transfer and updating of machine learning models.

Acumos provides a design studio that makes it easy to chain together a series of components, by connecting data sources to model-based predictors and then connecting those predictors to adapters that actually operate equipment, or provide other common functions like filling in spreadsheets or performing any developer-defined steps to create a simple and efficient decision tree. Such a design tool is not well suited to developing the actual machine learning algorithm. It is not intended to handle loops and recursive operations, but it is a tool that makes it extremely easy to integrate well-defined and independent microservices into powerful IT solutions.

STEP 4 - DEPLOYING FINISHED SOLUTIONS TO A RUNTIME ENVIRONMENT

The final step in building a working AI solution, as shown in Figure 1, is

to deploy a working application into a runtime system. In general, AI solutions will not run best in a centralized environment. For instance, it is hard to imagine an autonomous vehicle making an HTTP request to a central brain to make real-time decisions like avoiding pedestrians. In most cases, a good AI solution will have to be packaged and delivered to a runtime system where it will actually be used. Acumos packages solutions into Docker image files which can then be deployed into any Docker environment and managed through a set of container management tools, such as Kubernetes. Docker containers are a useful tool for deploying software, but other packaging and management systems exist today and, no doubt, others will proliferate over time. So the basic design of packaging solutions and deploying them will be adapted to any container-like mechanism.

Acumos provides tools to package any set of components, including predictors, adapters, and other microservices, as needed, to any runtime environment and to create a compatible, deployable image file. Such image files can be deployed to Azure, AWS or other popular cloud services, to any corporate data center or any real-time environment as long as it supports Docker or other supported lifecycle management tool. The adaptive and programmable deployment interface allows Acumos to package and transfer runtime bundles to a wide array of external systems.

A COMMON OPEN SOURCE AI FRAMEWORK

Acumos is pursuing an open source approach in order to establish a common framework that can be adapted to any existing or future AI development tools, databases and execution environments. A developer who wishes to use Acumos simply downloads the framework from ***acumos.org*** and installs it on their own environment, such as a desktop,

datacenter or cloud service of their choice. This is called an ***Acumos Instance***.

If the user is a modeler employing a supported toolkit or library, they can then utilize the Acumos onboarding libraries to take solutions they have built and store it on their Acumos instance. If the framework does not already include a library for onboarding from your toolkit of choice, the library source code is available to be adapted in order to integrate virtually any SDK or software design tool with the platform.

Once stored on an Acumos platform, onboarded code is converted to a ***microservice*** and stored in a local repository. Acumos supports Nexus natively but can be extended to a variety of repository implementations, all managed through a common data access component. Models can then be published either as source code or as a signed executable, to a catalog. Catalogs can be searched through the Acumos marketplace portal by users on the local instance, or the catalog can be shared with other instances in a pre-defined peer community.

Models can be acquired by members for the community, trained on problem-specific data and chained together with other components to create a solution. Finished solutions, in turn, can be deployed to a target runtime environment. So, whether the solution is intended to run in a cloud, a corporate datacenter or an independent system such as a webcam, robot or autonomous vehicle of some kind, the Acumos packager can be adapted, with the proper programming, to deliver solution packages to that target environment.

A DISTRIBUTED MARKETPLACE

Acumos implements a distributed AI marketplace in which individual Acumos instances exchange solutions with each other or via a general public intermediary so that solutions built by one company can be shared, licensed or sold to other companies. This substantially lowers the barrier

to entry for AI software in the way that many online markets simplify the process of exchanging goods or software. The marketplace makes it easy to find AI solutions that have been built, tested and used by others and makes it possible for data scientists to market their own innovations to others in an automated way.

Each Acumos instance may join with other instances to form a marketplace which shares a marketplace catalog. Instances may be in one marketplace, or several. Individual microservices can be published from an Acumos instance to one of the ***marketplace catalogs*** offering that microservice to users in that marketplace. The catalog allows users to examine, compare, discuss and explore components developed by programmers on different instances in the marketplace. Once a component is selected from the catalog, it can be ordered from its owner. The catalog also allows component owners to track usage, issue updates and answer queries about their software.

This highly distributed, federated model provides more flexibility than a central store and allows proprietary components to be published into specialized catalogs with limited distribution. One can imagine that a company can federate to a catalog of only internally developed or approved components without exposing users to any external, unsecure components which may be in public catalogs. A well-known ***public*** marketplace, ***acumos.org***, can provide an easy to access central catalog of open, commercial AI models. And between these two extremes, individual restricted communities, or communities of common interest can be formed around quasi-public, privately managed marketplace systems with their own policies on acceptance and sourcing of components.

INTEROPERABLE MICROSERVICES

Acumos provides a packaging function to wrap individual components, including models, adapters and public data sources into microservices making it simple to build composite solutions that operate on common data, even when each component of a composite solution is built using a different language or with a different toolkit or SDK. Acumos microservices are always interoperable because they are built to implement a common communications model and are run in independent containers.

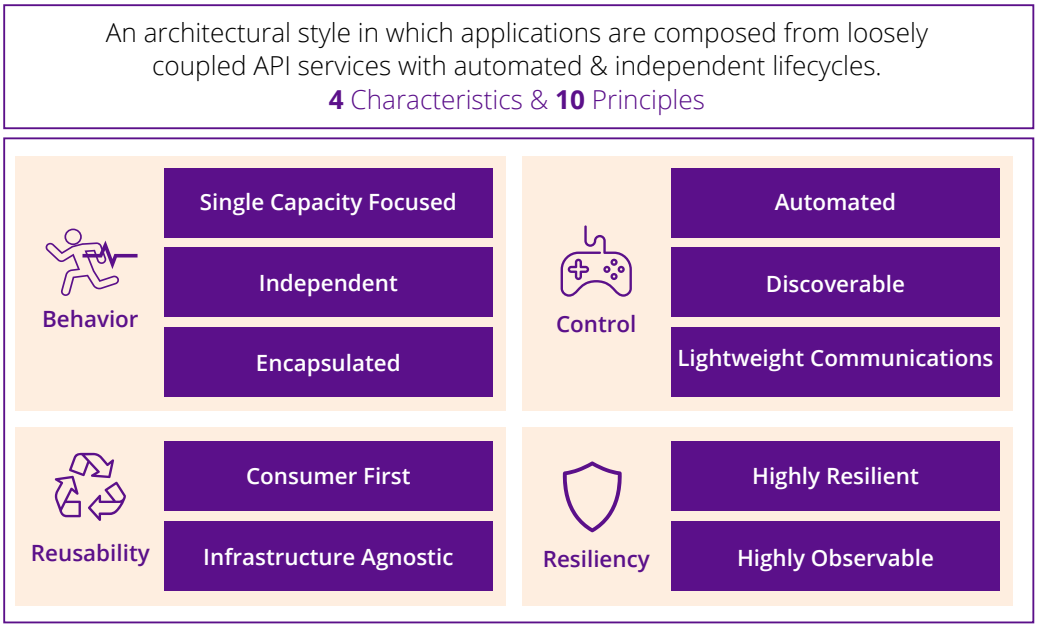

Figure 2. The characteristics of microservice

A microservice is a component that follows a specific architectural style of building business functions from small units of reusable capability (see Figure 2). A discrete application can be composed of a loosely coupled set of such microservices, where each has a specific function or role in an overall use case, built without any external dependencies on other microservices, so that it can be replaced at any time with a different implementation, and each has a well-defined API and an automated and resilient lifecycle. Thus, each microservice is independently restarted on any failure and can be replaced, with no impact on other microservices

or on any composite solution built from these microservices. The goal is to encapsulate a single function, create modularity, without adding complexity to the application which it supports. In the case of Acumos, these microservices are composed of Docker containers making them simple to deploy and manage on a wide range of target environments. If microservices are extremely specialized, they cannot be easily integrated or deployed.

Using the Acumos design studio, individual microservices are combined, through a simple chaining process, to form integrated runtime solutions and training packages. Each is stored with its own TOSCA component blueprint so that the API for invoking the module is explicitly provided in a machine-readable form and all dependencies are documented, making automated assembly of composite solutions a straightforward process. The microservices are then combined, as part of a larger solution in a Docker image file, a packaging construct which makes them easy to relocate to a wide range of target runtime environments on which they can be executed.

This paradigm for assembling models into runtime solutions or for training packages, maximizes the reusability of machine learning models. It allows data scientists to develop code which can be trained by others without the need to manually integrate the model into an application using a coding process, or to alter the model for compatibility with a particular runtime. Thus, modelers can deliver debugged microservices and trainers can use them without the need to know a great deal about the internal characteristics of the model they are training and virtually nothing about the data science tools which were used to create it.

Because of the independence of microservices from one another, very complex functionality can be built through a composite design. An example service might take a model that detects faces and pass the location of each face to another model which then operates only on the faces, for instance, by blurring faces for anonymity (as shown in Figure 3) or recognizing each of the individual faces detected by the previous microservice.

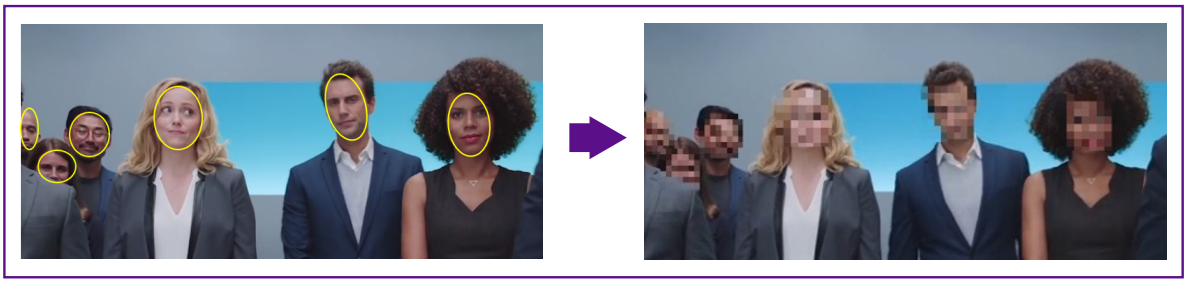

Figure 3. Chained microservices for video processing

Microservices make it possible to incorporate individual components into complex processing sequences that may be required to accomplish very complex tasks without the need to change the code of any individual module.

AN ECOSYSTEM TO EXPEDITE INNOVATION

Since Acumos solutions are always interoperable, documented by a catalog, transferrable using the Acumos publication process, discussed, rated and reviewed, and maintained through a consistent update process, it is easy to establish a community of interest that can work on common problems or technologies. Members of such a community can work individually on their problems, but by working together in an ecosystem, they can expedite innovation and introduce new technologies much more quickly. This ecosystem of AI solutions and machine learning tools means that developers are not on their own to provide an end-to-end solution for each use case. Data scientists can design and code new algorithms for data analysis, while IT developers can independently make use of the latest algorithms to solve their commercial problems. At the same time, technology specialists can train these algorithms on their own data in a private and secure environment.

The goal of the ecosystem is to create an environment which simplifies the exchange and distribution of software in order to lower the barriers

to entry for machine learning solutions. Acumos attempts to complement existing AI tools, rather than displacing them, in order to broaden the audience and reduce the training necessary to utilize solutions offered by the many technologies which already exist. At the same time, we seek to future-proof this rapidly evolving technology by ensuring that new toolkits and techniques can be integrated with existing solutions. It is necessary to create a technology-neutral platform in order to build confidence in machine learning and increase exposure to a wider market.

The barriers are further reduced because these marketplaces can be used to report software issues and repair them and, by following a chain of custody of the microservices and the applications in which they have been deployed, automate the software update distribution process, as well. An ecosystem is created because software developers and users are able to cooperatively improve solutions based upon their experience with any software which enters the marketplace.

6 Acumos Design Studio and the Integration of AI Models

The use of machine learning to drive AI is a very useful approach to some problems, but it doesn't replace conventional programming in every respect. Rather, machine learning is a tool for addressing a class of problems. One type of problem that machine learning is best applied to is the general classification problem. Given an image, can we classify the contents to determine what this is an image of?

Figure 4. Building complex video services from modular components

In reality, most images are not limited to a single subject. If a classifier can learn to detect faces, we can list all of the faces in an image. So for a limited set of problems – e.g. group all the photographs of Uncle Fred in my collection – that may be all we need to know. But, in most practical use cases, the challenge is much greater. In Figure 4 and the following section, we discuss what it takes to assemble a number of independent attributes taken from different data sources, just to create an application that assembles a single, customized video feed.

INDIVIDUALIZED MODULAR COMPONENTS

Figure 4 shows an example of a single video segment and the many distinct features and data sources that are part of each frame of that video segment. If I were given the task to create a curated video feed in which I want to produce a sequence of such video segments to a consumer based upon the kind of programming that consumer prefers, there may be many separate determinations that must be made.

- For example, I may need to compose a 30 minute program, constructed from a collection of video segments and ads. So, I have to detect the individual segments in a set of candidate program sources, such as the network news programs for the day, and select the ones that add up to 30 minutes of content.
- Because I know the attention span of the user, I may want to constrain segments so that no single segment is more than 4 minutes in length. Length is a feature of a video segment, so I must be able to detect the beginning and end of each segment, which might require, for instance, that we detect fades or wipes in the video stream.
- Furthermore, I may want to classify all the available segments to find the topics that most interest the user, such as science related news segments, or local politics, for instance. This is information which may best be determined by examining closed caption data containing phrases that can quickly classify the segment with respect to topic.
- Among the segments being considered, there may be some individuals that are more popular with my user than others. This might be best determined by examining individual frames prominently displaying faces and applying a face recognition algorithm which has been trained specifically on the viewer's

favorite celebrities, scientists, correspondents or sports figures.

- Furthermore, in order to best monetize this curated feed, I may want to identify products, logos or locations that appear in individual shots, in order to select the ads which, when placed between segments, will maximize the advertising value of the program.

Trying to build a single AI program which can accomplish all of these tasks, and perhaps several others, at once is quite impractical. An application which looks at program schedules, video segments, closed caption content, audio tracks, and individual images, looking for people, places, graphics and products will be very complex. Developing such a program, requires a tremendous amount of time and even though we want to use machine learning to drive this decision-making process, training that program to recognize so many factors will be completely impractical. It will be difficult to acquire all the necessary data and, because it will only be able to accomplish that one job, any future enhancements or changes will completely invalidate the entire process.

In many ways, machine learning algorithms have similar issues to the familiar constraints of training human beings. There is a significant investment required in educating the individual, so it is no more practical to teach one monolithic program many specialized tasks at once than it would be to train an individual in multiple specialties. If you were establishing a project involving humans, you would never train a person in medicine, and finance, and engineering just because those skills were all needed for the project. Instead, in the real world, one would assemble a team of people who, together have all the necessary skills.

Even if it were argued that the total training time is the same, it is more efficient to add or remove individuals from a team when different skills are needed. A well-integrated team is a better approach to solving a problem because the individual members of the team can go on to solve other problems, in the future, by moving to other teams with a different mix of skills. So over several projects there is less waste.

Microservices make it possible to apply the same technique to the construction of AI software. In our curated video use case, it is much more logical and efficient to build and train a series of individual microservices, each with their own specialized algorithm and each looking at different data sources and combining them into a full solution. In Acumos, we call this a composite solution as it is made up from separate, independently trained microservices which are then assembled together into a single decision tree to form a completed application.

COMPOSITE APPLICATIONS

Video Analsis: Separate, composable microservices assembled in combination for a desired outcome (e.g. ad placement, video curation, viewing preferences)

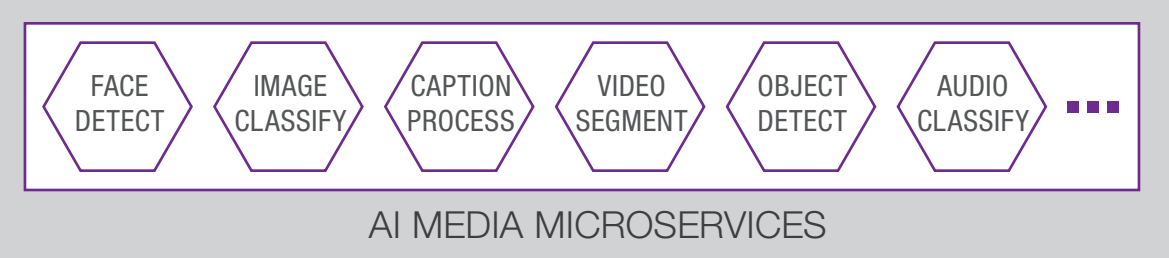

Figure 5. Some example microservices

Acumos provides a design studio which allows individual microservices, whether they encapsulate AI models, data transformation algorithms or data acquisition components, to be chained together to form a composite program. Essentially, any microservice components can be used to form any composite program which, in turn, is packaged by the design studio as another self-contained microservice. The design studio uses component models, or TOSCA blueprints, to identify the input-output requirements and package dependencies of its microservices, making it very simple to integrate a wide variety of individual components into a single composite program.

Design studio is a tool that helps developers quickly assemble these composite applications using intuitive composition mechanisms.

The design studio tool is based on a novel and flexible approach to building domain-specific machine learning applications. The individual microservices are building blocks presented on a palette which can be integrated based on **requirements** exposed by one component and **capabilities** offered by another. Each of these attributes are determined when individual models are onboarded to Acumos and stored in the component blueprint, in TOSCA format. Design studio does not provide the logic, it simply identifies the process flow for a finished application, so it doesn't compete with other SDKs that are used to design the individual components.

PACKAGING AND DEPLOYMENT

Once a solution is designed, with all of the component microservices available in the repository, it can be assembled into a Docker image file and deployed to a target runtime environment. Acumos does not incorporate its own execution environment. Instead, when a user is ready to deploy a solution, the user selects one of the supported target environments or services. Acumos supports a variety of deployment API's, for instance for deployment to an Azure cloud or to an OpenStack platform. At deployment time, Acumos packages the designed soultion, along with the appropriate runtime libraries into a deployment file which can be installed directly into a cloud service by leveraging the user's tenant credentials, or downloaded to a local machine as a bundle which can be separately delivered and deployed into any Docker-compliant environment.

This packaging process allows a target artifact to be created at deployment time to meet the needs of the specific runtime environment. This provides a flexible process that can be used for delivering either a training bundle or a finished service. Such a packaging and deployment capability makes it possible to run Acumos-designed solutions on any cloud platform, including platforms with specialized hardware acceleration.

It also allows Acumos-designed solutions to be deployed in specialized environments not connected to any cloud service.

As an open-source platform, the Acumos packaging and deployment process can be adapted to additional runtimes, as needed, either by the opensource community or, individually, by users.

7 Summary

The goal of Acumos is to make AI and machine learning accessible to a wide audience by creating an extensible marketplace of reusable solutions, sourced from a variety of AI toolkits and languages that ordinary developers, who are not machine-learning experts or data scientists, can easily use to create their own applications.

Such a generic AI platform is needed to meet the demands of a broad range of business use cases, including network analytics, customer care, field service and equipment repair, health care analytics, election results auditing, network security and advanced video services, to name just a few. By making the platform open-source, we leverage the development power of many partners and participants, leading to greater velocity in moving the platform forward. This will enable the creation of solutions to real business problems more quickly and cheaply, resulting in tangible and substantial business value. Without early demonstrations of business value, such technologies are unlikely to become mainstream.

Acumos can help scale up the introduction of AI-based software across a wide range of industrial and commercial problems in order to reach a critical mass of applications. In this way, Acumos drives toward a data-centric process for producing software, with AI and machine learning as the central paradigm. The platform empowers data scientists to publish adaptive AI models without forcing them to engage in the custom development of fully integrated solutions using those models. Ideally, software developers will use Acumos to change the process of software development from a code-writing and editing exercise into a classroom-like code training process. Acumos focuses on training models after they are onboarded, published and delivered to a user, rather than as part of the model development process, so that access to training data becomes more secure and more convenient. This offline training involves both the construction of training datasets and scoring, the evaluation of trained models on their ability to successfully analyze test datasets that are fed to them. End-users can then select the best model for the job, and integrate that model into a complete application. The containerized microservices

built by Acumos fully encapsulate the AI models, insulating ordinary developers from their complexities. Acumos ensures interoperability with other supported components, including other models and input/output adapters which Acumos can chain together, independent of their source languages, libraries or the tools with which the components were originally built.

Acumos is not a centralized execution environment for AI solutions. It is a design and distribution framework for integrating solutions from modular components. It provides a launchpad for training and validating both individual components and integrated, or composite, solutions and then securely distributing the results to targeted communities through an electronic catalog from which components can be selected and licensed. Acumos also provides the deployment interfaces which allow solutions to be trained or executed on many popular runtime environments, including several commercial cloud services.

Acumos is not tied to any one run-time infrastructure or cloud service. It supports many hardware infrastructures in order to maximize the utility of the solutions being deployed. This makes Acumos-compatible solutions portable and flexible. Acumos offers a mechanism for packaging, sharing, licensing, and deploying AI models in the form of portable, containerized microservices, which are interoperable with one another. It provides a publication mechanism for creating shared, secure catalogs and a mechanism for deployment onto any suitable runtime infrastructure. Acumos works in conjunction with popular tools, such as Docker, to manage the lifecycle of machine learning models and the applications that they power, on a broad set of execution platforms.

REFERENCES

McKinsey Global Institute. (2017, June). Artificial Intelligence The Next Digital Frontier? Retrieved from http://www.mckinsey.com/~/media/McKinsey/Industries/Advanced%20Electronics/Our%20Insights/How%20artificial%20intelligence%20can%20deliver%20real%20value%20to%20companies/MGI-Artificial-Intelligence-Discussion-paper.ashx

Silver, D. e. (2017). mastering the game of Go without human knowledge. Nature, 354-359.

Acumos AI is a platform an open source framework to simplify the development of Artificial Intelligence / Machine Learning.

To learn more about Acumos, please visit us at acumos.org.