Whitepaper 2.0

SingularityNET

A Decentralized, Open Market and Network for AIs

Whitepaper 2.0: February 2019

Abstract

Artificial intelligence is growing more valuable and powerful every year and will soon dominate the internet. Visionaries like Vernor Vinge and Ray Kurzweil have predicted that a "technological singularity" will occur during this century. The SingularityNET platform brings blockchain and AI together to create a new AI fabric that delivers superior practical AI functionality today while moving toward the fulfillment of Singularitarian artificial general intelligence visions.

Today's AI tools are fragmented by a closed development environment. Most are developed by one company and perform one extremely narrow task, and there is no straightforward, standard way to plug two tools together. SingularityNET aims to become the leading protocol for networking AI and machine learning tools to form highly effective applications across vertical markets and ultimately generate coordinated artificial general intelligence.

Most AI research today is controlled by a handful of corporations—those with the resources to fund development. Independent developers of AI tools have no readily available way to monetize their creations. Usually, their most lucrative option is to sell their tool to one of the big tech companies, leading to control of the technology becoming even more concentrated. SingularityNET's open-source protocol and collection of smart contracts are designed to address these problems. Developers can launch their AI tools on the network, where they can interoperate with other AIs and with paying users.

Not only does the SingularityNET platform give developers a commercial launchpad (much like app stores give mobile app developers an easy path to market), it also allows the AIs to interoperate, creating a more synergistic, broadly capable intelligence. For example, if a text-to-speech AI and an Italian-to-English translation AI were both on the network, then the network as a whole would be capable of using Italian text to produce English speech.

Within this framework, AI transforms from a corporate asset to a global commons; anyone can access AI tech or become a stakeholder in its development. Also, anyone can add an AI/machine learning service to SingularityNET for use by the network and receive network payment tokens in exchange.

SingularityNET is backed by the SingularityNET Foundation (described in section 1.5), which believes that the benefits of AI should not accrue only to a small set of powerful institutions, but rather should be shared by all. A key goal of SingularityNET is ensuring that the technology is benevolent by human standards and that the network is designed to incentivize and reward beneficial players. This is critical given the explicit aspiration of the project—alongside shorter-term practical and commercial goals—to play a central role in launching

the technological singularity as foreseen by Vinge, Kurzweil, and others by catalyzing the emergence and economic pervasiveness of self-modifying, self-improving, self-understanding artificial general intelligence.

The SingularityNET platform, AI network, and ecosystem are works in progress, and are intended to remain so, the very essence of the singularity being rapid change. In this spirit, what you are reading is a substantial revision of SingularityNET's first whitepaper. The first version was written in fall 2017 before the network's initial token generation event, whereas this version was written in February 2019 and reflects what has been learned and built during the interim.

Contents

1. Vision

1.1 Inspiration

The inevitability of a technological singularity is increasingly accepted throughout the technology and business worlds. Knowledgeable people are realizing that the next few decades will see a transition to a new society and economy in which machine intelligence is the dominant factor. For this to occur, swarms of machine and organic intelligences must network together to produce emergent "global brain" dynamics of unprecedented complexity and sophistication with power and flexibility none alone would have.[1]

Markets display elements of cognitive synergy—agents in an economy each pursue their own relatively simple goals, but patterns with higher-order goals emerge from their interactions. SingularityNET is not only a collection of AIs, it is a market. It is designed to harness self-organizing swarm intelligence to create a whole greater than the sum of its parts. Blockchain allows us to program economic rules in a digital environment and AI software to seamlessly interact with them.

The path to creating a positive "global brain" is challenging. A technological singularity could have unprecedented benefits but also poses unprecedented risk. The popular press is full of dire warnings about the dangers of artificial general intelligence.

Among the challenges is the current set of protocols for collective action; in many respects, today's financial mechanisms and institutions would give us a risky ride to the singularity. New, more flexible, open, and rapidly adaptive economic structures and dynamics are needed.[2]

Blockchain, with its natively digital money, is a powerful tool for managing transactions in an economy dominated by machine intelligence.[3] However, blockchain is just a tool; there are important decisions to be made about how to use it. SingularityNET is a blockchain-based framework designed to serve the needs of AI agents as they interact with each other and with external customers. At its core, SingularityNET is a set of smart contract templates that AI agents can use to request that AI work be done, to exchange data, and to supply the results of AI work.

This framework is a network that meshes disparate elements into a collective intelligence, much like the different areas of the brain—each with its own speciality—mesh together. It is critical that this network be designed with positive principles in mind:

- Democratic governance on specific issues—if the community governs the system, then the system will tend to act for the benefit of the community

[1] Damien Broderick, *The Spike* (Tor Books, 1997); Ray Kurzweil, *The Singularity Is Near* (2006); Vernor Vinge, "The Coming Technological Singularity," *Whole Earth Review* 81 (1993): 88–95; Ben Goertzel, "Human-level Artificial General Intelligence and the Possibility of a Technological Singularity: A Reaction to Ray Kurzweil's 'The Singularity Is Near,' and McDermott's Critique of Kurzweil," *Artificial Intelligence* 171, no. 18 (2007): 1161–73.

[2] Ben Goertzel, Ted Goertzel, and Zarathustra Goertzel, "The Global Brain and the Emerging Economy of Abundance: Mutualism, Open Collaboration, Exchange Networks and the Automated Commons," *Technological Forecasting and Social Change* 114C (2016): 65–73. 2016, http://goertzel.org/OpenCollaboration.pdf.

[3] John Clippinger and David Bollier, *From Bitcoin to Burning Man and Beyond* (Off the Common Books, 2014).

- Encouragement that prompts innovative new agents to enter the network, and creation of the conditions necessary for agents to act in a manner that feeds the collective intelligence

- Direction of a significant percentage of the network's efforts toward causes of broad benefit

SingularityNET has been designed to meet these requirements by

- delivering intelligence services to corporations, other organizations, and individuals;

- fostering the emergence of increasingly powerful distributed general intelligence; and

- deploying artificial intelligence for ever-increasing benefit to as many humans and other sentient beings as possible.

SingularityNET is designed both to be highly valuable now and to lay the groundwork for the emergence of a self-modifying, decentralized "artificial cognitive organism" with the eventual potential for general intelligence and beneficial ethical characteristics beyond the human level. It is a practical design inspired by long theoretical thinking and prototyping by our founders regarding artificial general intelligence,[4] open-ended intelligence,[5] the global brain,[6] and more.

1.2 Acute Market Needs Addressed

SingularityNET meets an acute and accelerating market need. In the current economic and technological context, every business needs AI, but off-the-shelf AIs will rarely match a business's needs. Only tech giants can hire armies of developers to build custom AIs, and even they have a hard time hiring enough AI experts to meet demand. SingularityNET provides an automated process that enables any business to connect existing AI tools to build the solution it needs. It optimizes for accessibility and customizability and by its nature reduces the reduplication of effort involved in proprietary development, making development more efficient.

Many state-of-the-art AI tools exist only in GitHub repositories created by graduate students or independent researchers. The latest algorithms for image and video analysis, machine translation, automated theorem proving, bioinformatics data analysis, etc. are typically available on Github, but the friction inherent in installing, configuring, and running them limits their use. Many remain little more than demos. Most AI developers are academics, not businesspeople, and have no easily accessible marketplace to turn to in order to monetize their clever AI code. As a result, the AI in real-world products tends to lag months to years behind the code.

[4] Ben Goertzel, *The AGI Revolution* (Humanity+ Press, 2016).
[5] David Weaver Weinbaum and Viktoras Veitas, "Open-ended Intelligence," *in International Conference on Artificial General Intelligence* (Springer, 2016): 43–52, https://arxiv.org/abs/1505.06366.
[6] F. Heylighen, "The Global Superorganism: "An Evolutionary Cybernetic Model of the Emerging Network Society," *Social Evolution and History* 6, no. 1 (2007).

SingularityNET is a launchpad where developers can quickly get their AI models and algorithms into real-world applications.

Machine learning tools also require datasets of sufficient size. Creating and managing such large datasets are beyond the means and capabilities of most AI developers, and the closed development model that currently prevails makes it hard for developers to share datasets.

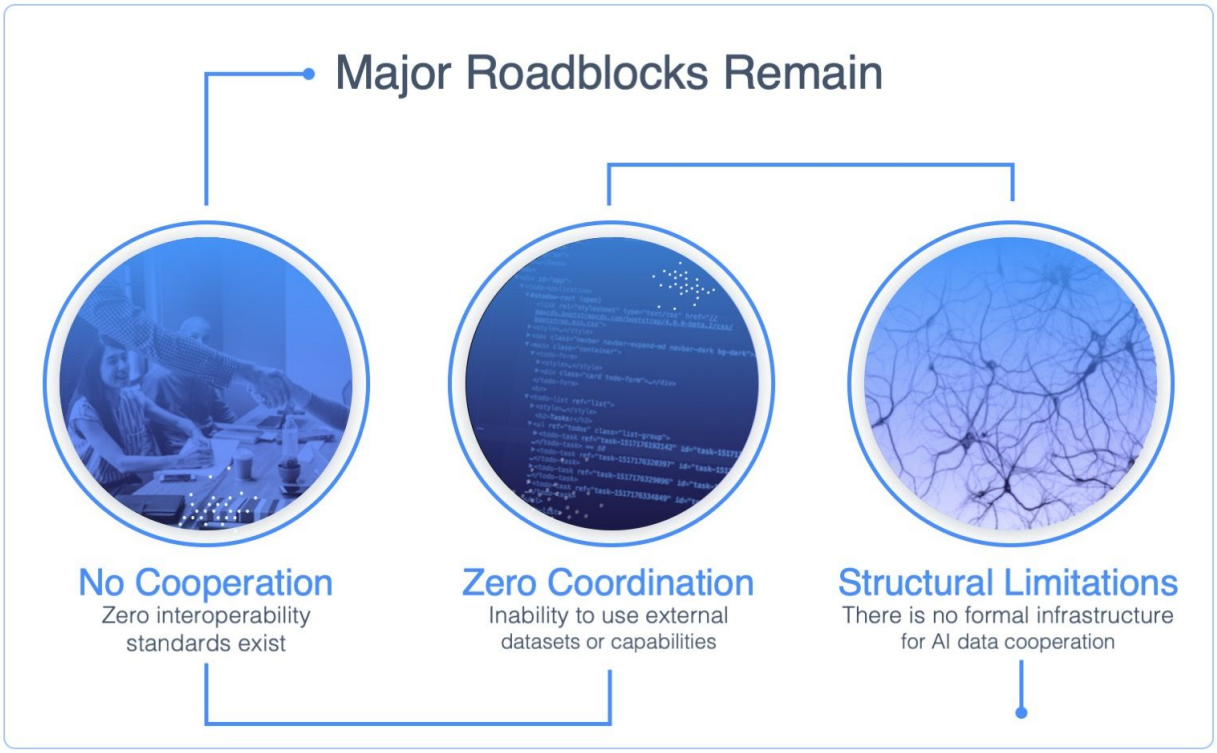

Figure 1. Major roadblocks remain

SingularityNET connects these AI tools and datasets to the marketplace, making them accessible to end users and developers and giving developers a way to monetize their creations. It is a sharing-economy marketplace for AI, allowing these tools to share data and abilities in order to democratize access to the benefits of AI.

In accordance with these goals, SingularityNET will be an open network. Anyone can insert an AI Agent as long as it shares information according to the SingularityNET API and accepts/disburses payment according to SingularityNET's economic logic. New AI Agents will come from AI developers who want access to SingularityNET's marketplace and who want to boost their AI agent's intelligence by linking it to other AIs in a cooperative network.

Like Uber and Airbnb, we have identified a large unexploited resource and a large market in need of that resource, and we are launching a tool to connect the two. The unexploited resource is AI algorithms and software on GitHub and elsewhere, and the market is the 99 percent of businesses that cannot afford a team of AI experts.

But in SingularityNET we also have a key added factor not present in these analogous cases: the apartments in AirBnB's network do not combine to become meta-apartments, nor does Uber's network create meta-cars, but AIs in SingularityNET's network come together to form meta-AIs whose intelligence is more than the sums of their parts. An unprecedented combination of powerful network effects is here, waiting to kick in once the network of AIs and associated human communities reach sufficient size and maturity.

1.3 A Robust and Adaptive Software Architecture

In computer science terms, SingularityNET is a distributed computing architecture for making new kinds of smart contracts to facilitate market interactions with AI and machine learning tools. The following design principles are incorporated throughout the design:

- *Interoperability*. The network will be able to interface with multiple blockchains.
- *Data sovereignty and privacy*. The network includes user-side controls for sharing personal data. Users remain in control of their data and can share it with the network via smart contracts.
- *Modularity*. Flexible network capabilities make it possible to create custom topologies, AI Agent collaborations of arbitrary complexity, and failure recovery methods.
- *Scalability*. SingularityNET will securely host both private and public contracts, allowing more scalable and resilient applications to be built with near-zero transaction costs.

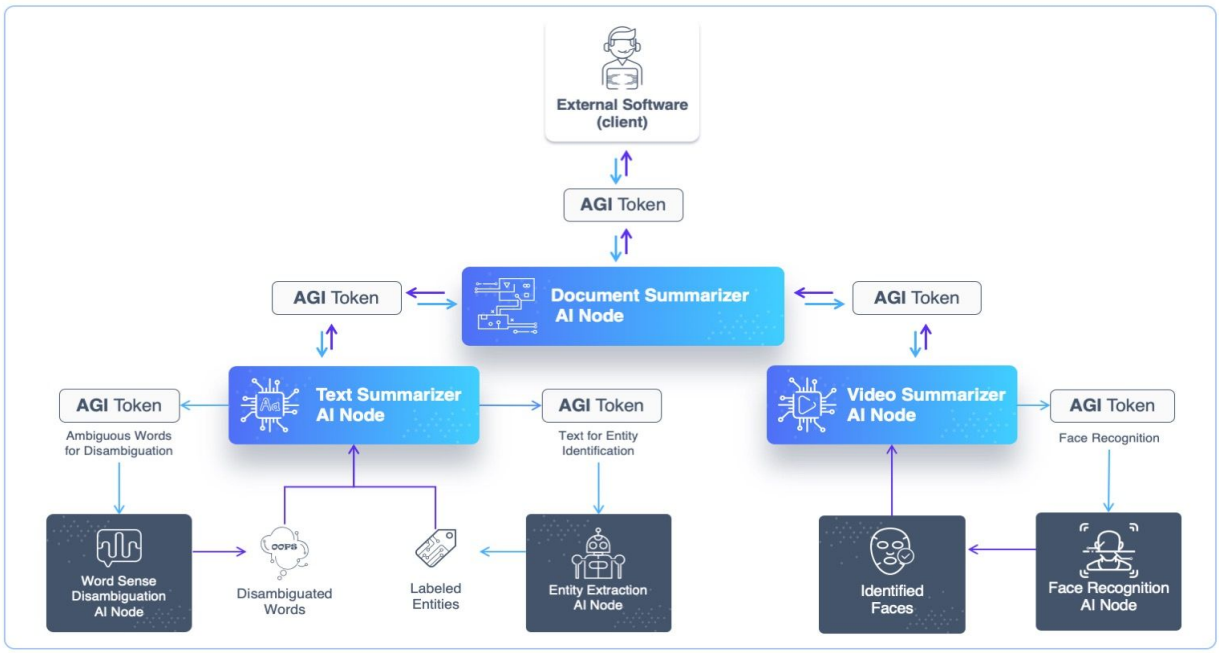

Figure 2. Cognitive synergy between AI Agents

SingularityNET Agents can run in the cloud and on phones, robots, and embedded devices.

1.4 A Decentralized, Self-Organizing Cooperative

One can think about SingularityNET as a "decentralized self-organizing cooperative." This concept is similar to to the better-known "decentralized autonomous organization" ("DAO") but different in that a foundation will provide high-level oversight of SingularityNET. Our intent is that over time, the network will evolve into a truly decentralized and autonomous organization. This sort of organization will differ from an ordinary corporation by, above all, its openness.

SingularityNET's collection of smart contracts includes contracts to be used by external, non-AI Agents who wish to obtain AI services from Agents in the network. Anyone can create a node (an AI Agent), put it online (running on a server, home computer, robot, or embedded device), and enter it into the network so that it can request and/or fulfill AI tasks in interaction with other nodes and engage in economic transactions.

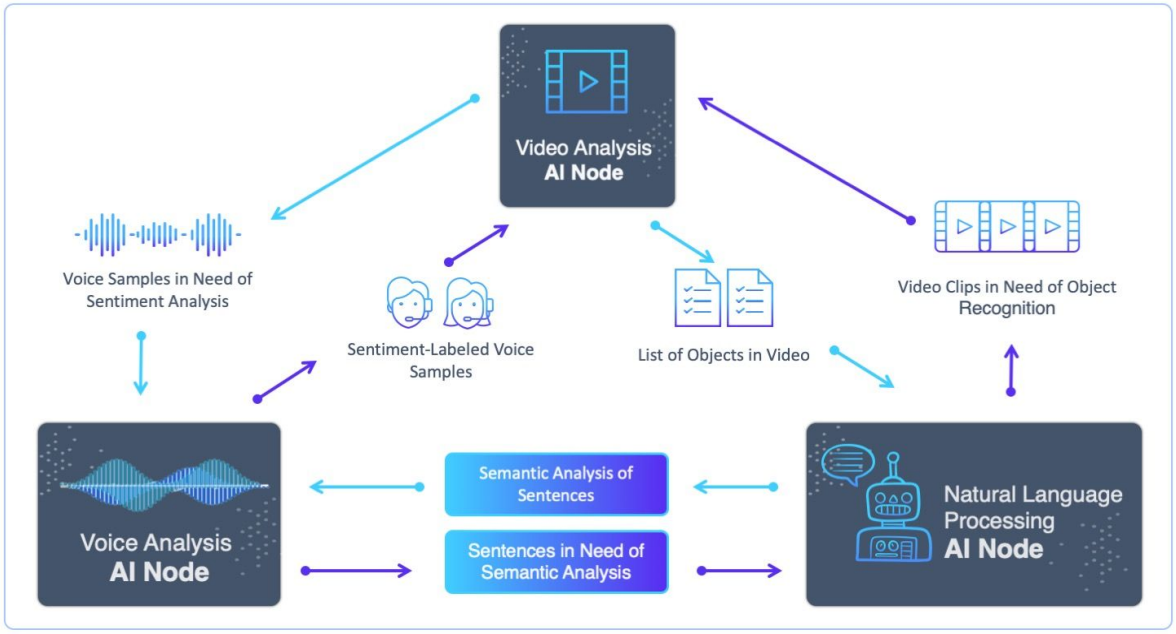

Figure 3. Example of a "circle of exchange" among AI Agents

Services on SingularityNET can be accessed using the AGI token. Token holders can use their tokens to purchase services in the marketplace. In the future, tokens may also bestow voting rights in the network's democratic governance system.

During the initial phases of the network's operation, the core parameters of the network will be governed by a nonprofit foundation which will be monitored and advised by a supervisory board. The foundation will operate the network and exercise oversight to prevent abuse and

hostile behavior while respecting the designed-into-the-system privacy of inter-agent interactions.

However, even in the earliest stages, activity on SingularityNET will be self-organized. For example, Agents will be free to create new AI Agents, insert them into the network, and transact freely and permissionlessly with each other.

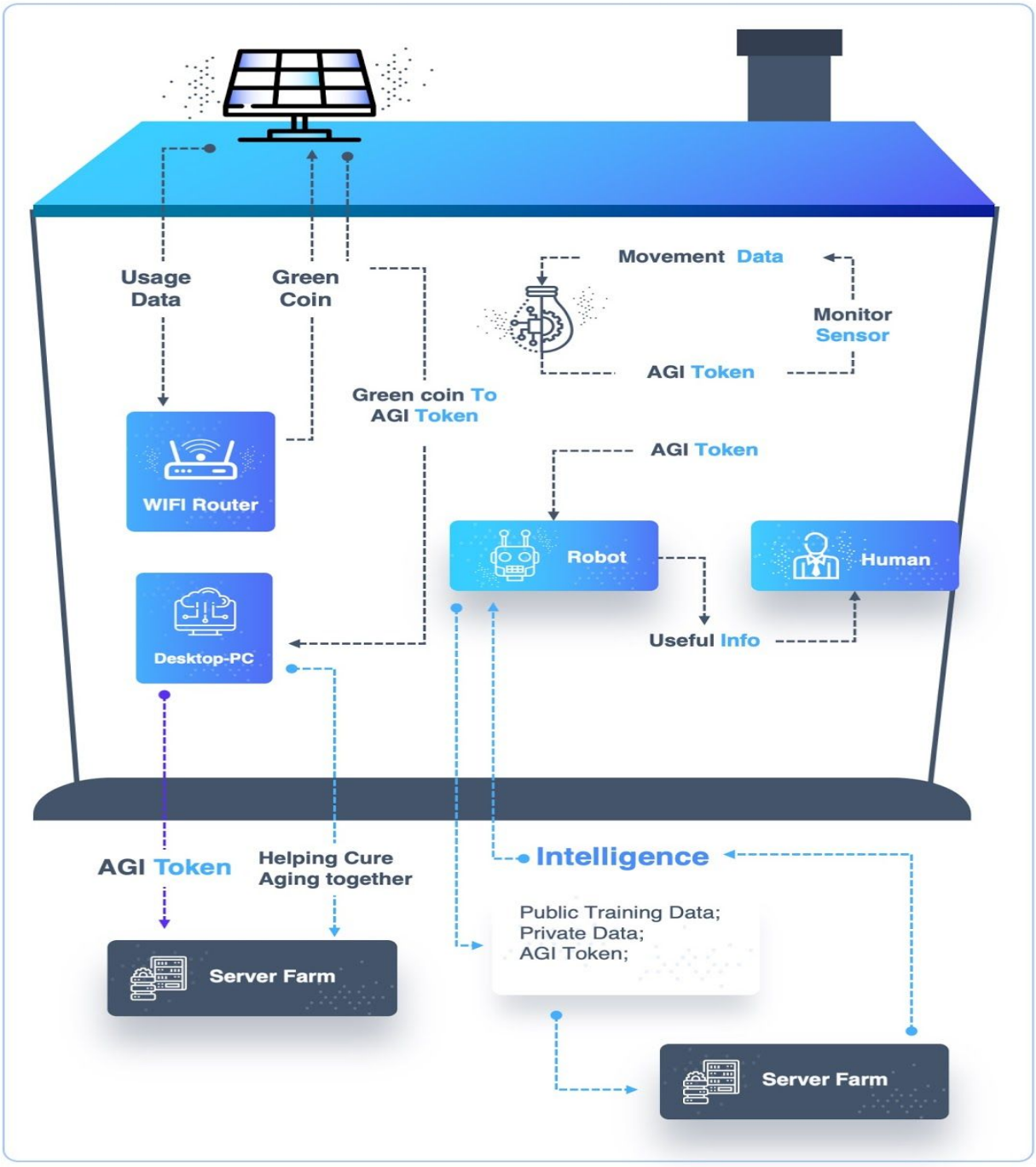

Figure 4. SingularityNET aims to foster intelligent systems and maximize their positive impact

SingularityNET aims to foster increasingly intelligent systems while maximizing the positive impact of these systems. The economic logic is designed to generate an intelligent global economy that pursues maximum benefits for all people and all life. Through a combination of powerful AI Agents, human decision-making, and benefit-maximizing game rules, SingularityNET will accelerate the development of a global supermind, helping humanity evolve into a more advanced, intelligent, beneficial, and connected mode of being.

In short, SingularityNET is an innovative economic mechanism designed to catalyze human and machine intelligence to move toward a new form of ethically beneficial self-organizing intelligence. Its global network of artificially intelligent agents will provide valuable AI services to anyone while, in the process, it self-organizes toward loftier goals. It is plausible that a highly successful SingularityNET will play a major role in the transition of humanity to a positive technological singularity.

The growth of SingularityNET will foster advances not only in practical narrow AI, but also in the general theory and practice of beneficial artificial general intelligence, in the design and analysis of structures for ethically intelligent economies, and in the continuous refining of means to conceptualize and estimate "benefit" and "greater good."

1.5 The SingularityNET Foundation

The non-profit SingularityNET Foundation, incorporated in the Netherlands, is responsible for building, supervising, and accelerating the growth of the SingularityNET network and marketplace.

During the initial phases of network operation, most major governance decisions will be made democratically by token holders, with the Foundation providing some high-level stewardship and practical day-to-day management. As the network evolves, there is potential for transition to a fully self-regulating Decentralized Autonomous Organization, and the network's technical specifications and governance methodology are designed to support this.

The SingularityNET Foundation was formed in late 2017 and early 2018 by the following key groups:

- The OpenCog Foundation, stewards of OpenCog, the leading open-source artificial general intelligence platform
- Hanson Robotics, creators of the world's most lifelike humanoid robots
- Vulpem, a blockchain software engineering consultancy responsible for back-end work on a number of successfully designed private and public blockchains, cryptocurrencies, and decentralized applications
- Artificial intelligence software consultancy Novamente LLC, which has provided custom AI solutions for corporations and government agencies since 2001

Creating a successful combination of sophisticated initial AI Agents, a flourishing community of AI Agent developers, and a rich ecosystem of customers at varying levels of sophistication is a huge undertaking. Fortunately, the founding team brought to the project significant experience as well as a large body of open-source code to help lay the foundation for the SingularityNET global brain network. But the vision requires the active participation of a grassroots community seeded by the founding team, both to put software on the network and to democratize governance.

2. The SingularityNET Platform

Realization of the SingularityNET vision requires a well-designed, efficient, and flexible underlying software platform that provides the protocols and tools necessary for AI agents to integrate into the network. Creating a platform that fully embodies the SingularityNET concept will be a medium-term effort, but significant progress has been made since development started in August 2017.

This section describes in moderate depth the software architecture for the beta version of the platform released in February 2019 and then makes some higher-level observations about features and improvements to be added post-beta.

2.1 Overview of the SingularityNET Beta Platform

The SingularityNET platform contains a number of critical components that work together to enable a decentralized network of AI services to flourish. The core components include many architectural components that allow for a functional, scalable, and extensible system. We arrived at this architecture through a careful process guided by a few key decisions governing blockchain interactions, AI service integration, and abstraction and by the goal of building an AI marketplace that is both open and compliant with regulatory and legal requirements.

First, we made the conscious choice to minimize our dependence on our current blockchain, Ethereum. Both conceptual and practical issues motivated this decision. Conceptually, we desire to be blockchain-agnostic and will consider building our own consensus algorithm based on reputation. The speed, reliability, and costs of Ethereum blockchain interactions dictate that any scalable system built on top of it must minimize gas costs and the delays introduced by block-mining time. These decisions are reflected in our use of tools to abstract away all blockchain interactions (the daemon, CLI, and SDK) and in our use of a multi-party escrow contract and atomic unidirectional channels for payments.

Second, on AI services integration, we wanted to abstract away as much of the network as possible, from an AI developer's perspective, in order to reduce the learning curve and minimize the overhead associated with providing AI services via the network. Moreover, we wanted to achieve this abstraction with a single flexible tool that also helps us provide scalability, robustness, and distribution and management features. This is achieved by the daemon, which is a sidecar proxy used to communicate with services and the network and which will soon also allow services to very easily find and call other services.

Finally, to make our marketplace compliant with regulations without compromising on openness, we implemented a fully decentralized registry of AI services available on the platform. The AI marketplace supplements that registry (a smart contract) with a centralized source for which services are curated; that is, they have gone through a due-diligence process covering the service owners and the nature of the service being offered.

The diagram below depicts the key components along with auxiliary components and their roles.

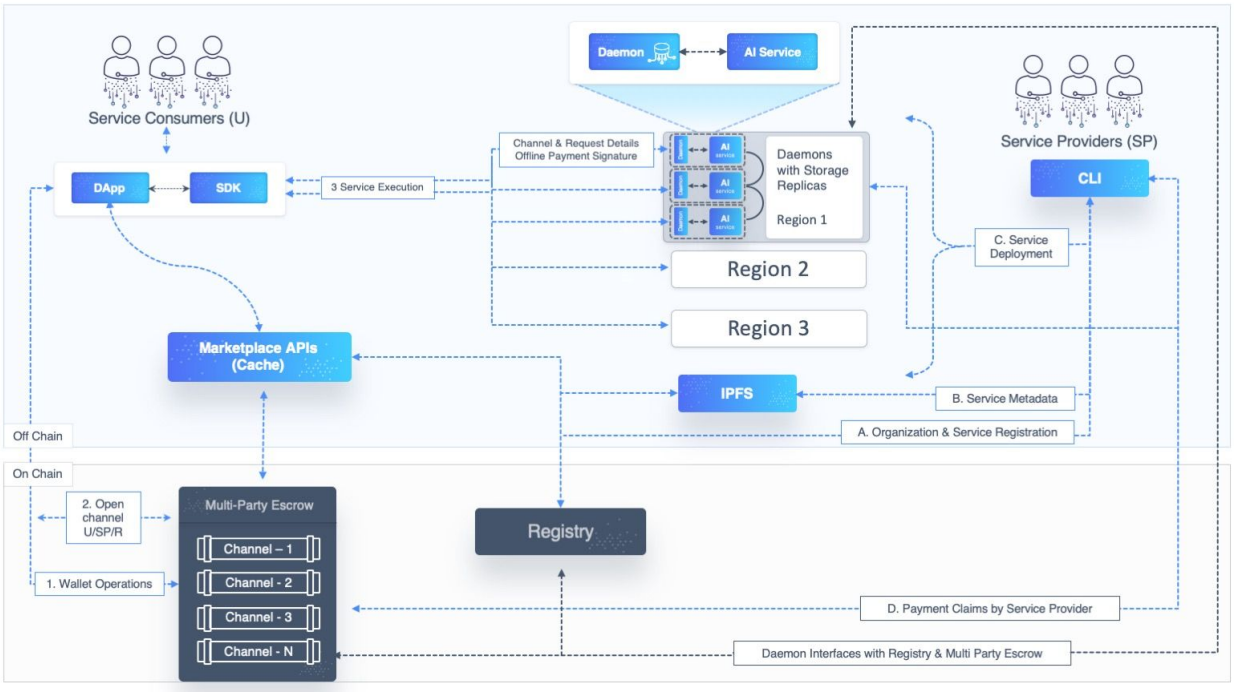

Figure 5. The key components of our platform

For a developer who wants to offer an AI service over the network, the most crucial component is the SingularityNET daemon, an adapter and proxy that abstracts away interactions will all other components. The daemon handles interactions with smart contracts and payments, takes care of client request validation, and does other useful tasks, allowing AI developers to focus almost exclusively on the AI-related aspects of their server-side software and services. The daemon is a sidecar proxy, so one daemon instance is deployed next to each AI service instance.

For end users who want to purchase access to the AI services available in the platform, the most important component is the **Marketplace DApp**, through which they can search and browse a collection of curated services (i.e., services approved by the SingularityNET Foundation as relevant and high quality and whose owners have signed user and data-privacy agreements) for a large and ever-growing variety of AI tasks. The Marketplace DApp also handles payment for services (through MetaMask integration) and service ratings.

For application developers who want to use the network's intelligence in their applications, the key component is the **SingularityNET SDK**, which automatically compiles client-side code

for interacting with the platform and with specific services, allowing service requests to be coded in a straightforward way and supporting payment and interactions with the blockchain.

The Ethereum blockchain is used to host two critical smart contracts: the **Registry** and the **Multi-Party Escrow**.

The Registry is where AI service providers register on the platform, which involves providing text descriptions and tags to allow users to discover their service, pricing information, and information such as gRPC models and endpoint locations to allow users to call their services.

The Multi-Party Escrow contract handles payments through escrow accounts for each user (end users and applications) coupled with **atomic unidirectional channels** for faster and cheaper transactions.

Those are the core components of the platform. Two key support components are also worth mentioning:

- The AI developer– and owner-oriented CLI (command-line package) provides command line APIs for a number of crucial service developer and service owner tasks: registering and managing identities, publishing services, updating registration information, notifying the platform of new endpoints, managing payment channels and balances, and calling services.

- The Request for AI Portal (RFAI) is a DApp through which end users and application developers can request specific AI services they would like added to the network and stake AGI tokens as a reward for high-quality solutions.

2.2 The SingularityNET Daemon and Wrapping Services

The daemon is the adapter that a service can use to interface with the SingularityNET platform. In software architecture lingo, the daemon is a sidecar proxy,[7]—a process deployed next to a core application (the AI service, in this case) to abstract away some architectural concerns such as logging and configuration as well as entire platform aspects, such as the interaction with smart contracts or even the decision to use the Ethereum blockchain.

The two key abstraction responsibilities of the daemon are payments and request translation. In order to authorize payments, the daemon interacts with the Multi-Party Escrow contract. Before invoking a service through SingularityNET, a consumer must have

1. funded the Multi-Party Escrow contract (see section on payments below) and

2. opened a payment channel with the recipient as specified by the service definition.

With each invocation the daemon checks that

1. the signature is authentic,

2. the payment channel has sufficient funds, and

[7] https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar.

3. the payment channel expiry is beyond a specified threshold (to ensure that the developer can claim the accrued funds).

After these successful checks, the request is proxied to the service. The daemon also keeps track of payment states of different clients.

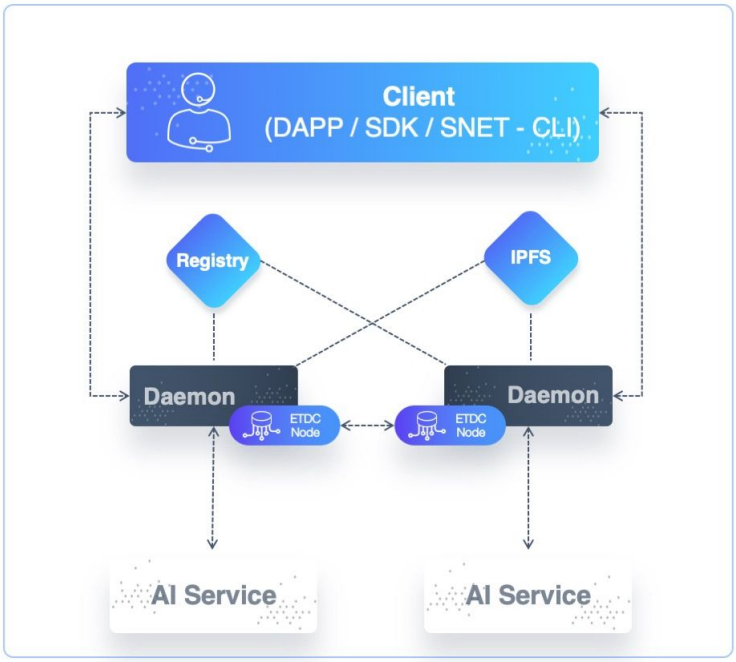

Figure 6. The SingularityNET daemon

Once the daemon has validated requests, it translates them into the format expected by the AI service. The daemon exposes a gRPC,[8] so all requests are based on gRPC and protocol buffers,[9] but it can translate requests to a few different formats, as expected by the service: in addition to gRPC/Protobuf, JSON-RPC and process fork–based services (executables to be executed on a per-call basis with the input parameters on standard input) are supported. This translation enables one consistent protocol to be used to communicate with any service on SingularityNET. The daemon and CLI also use gRPC and Protobuf for communication.

One can deploy multiple instances of an AI service. Each instance will have its own sidecar daemon, and all daemons will be registered as endpoints in the Registry. When multiple instances exist, they can be put into one or more *instance groups (*a typical reason for doing so would be to group instances in the same data center or cloud region). Daemons in the same group coordinate to share payment status information through etcd.[10]

The daemon provides some additional deployment- and administration-oriented features:

[8] https://grpc.io/.
[9] https://developers.google.com/protocol-buffers/.
[10] https://coreos.com/etcd/.

- *SSL termination*. This can be done either with a certificate and keyfile supplied by the service developer or with automatic certificates provided by Let's Encrypt.[11]
- *Logging to files, with log rotation and pluggable log hooks*. Currently an email hook is provided, and an easy-to-use API is available for other hooks.
- *Metrics, monitoring, and alerts*. The daemon collects metrics about request calls, which service owners can use to optimize their resource usage. It also monitors daemon and service events, providing configurable alerts via email or web services.
- *Rate limiting to prevent DoS attacks and to allow service owners to scale at their own speed and ability*. The daemon uses the token bucket algorithm.[12]
- *Heartbeat*. A pull-based heartbeat service is provided, following the gRPC health checking protocol.[13] The daemon will check that the heartbeat of the service is configured; this is used by monitoring services as well as the Marketplace DApp.

2.3 The SingularityNET Registry

The SingularityNET Registry is an ERC-165–compliant[14] smart contract on the Ethereum blockchain that stores organizations, services, and type repositories. AI developers use the Registry to announce details of their services, and consumers use the Registry to find the services they need. When a user searches for a service in the Marketplace DApp, the DApp reads details of the services from the Registry. The Registry also allows tagging of services and type repositories to enable searching and filtering.

The Registry stores four main pieces of data: organizations, services, type repositories, and tags. It supports creation, removal, editing, and reading for all of these, and contains several view functions for retrieving data.

An organization is an umbrella for services to be grouped under and is at the top of the Registry's data hierarchy. Service developers can (and should) register an organization and then put all of their services underneath it. An organization registration record has a name, an owner address (in the identity sense), a collection of member addresses, a collection of services, and a collection of type repositories.

Services and type repositories registered under a given organization are said to be owned by that organization. The list of members is a primitive access-management structure; members of an organization can do everything except change the organization owner and delete the organization.

A service represents a single AI service. Its Registry entry contains all the necessary information for a consumer to call that AI service. The entry contains a name, tags, and IPFS hash. The name is an identifier for discoverability, the tags help a customer find a service

[11] https://letsencrypt.org/.
[12] https://en.wikipedia.org/wiki/Token_bucket.
[13] https://github.com/grpc/grpc/blob/master/doc/health-checking.md.
[14] https://eips.ethereum.org/EIPS/eip-165.

without knowing its name, and the IPFS hash is the link to the metadata file on IPFS. DApps and smart contracts can use the listServicesForTag view function to discover services.

All service metadata is stored off-chain in IPFS for performance and gas-cost reasons. This metadata includes

- basic information such as version number, service name, description, etc.;
- code-level information for calling the service, such as encoding (protobuf or JSON) and request format (gRPC, JSON-RPC or process);
- A list of daemon endpoints, aggregated into one or more groups;
- pricing information; and
- an IPFS hash for the service API model.

The CLI provides a convenient API and library for manipulating this metadata.

A type repository is a Registry entry where a service developer lists service metadata, such as the service model and the data types used. The entry contains a name, some tags, a path, and a URI. The name and tags are for discoverability, the path is an optional identifier for the organization's internal management, and the URI allows the client (whether an end user or an application making calls through the SingularityNET SDK) to find the metadata. DApps and smart contracts can use the listTypeRepositoriesForTag view function to discover AI services. The URI is an IPFS hash, and the hosting itself can be done by either SingularityNET, the service developer, or any IPFS pinning service, such as Infura.

Tags are completely optional but recommended for discoverability. Registry functions allow tags to be added to services and to type repositories. After that, the tags are displayed and searchable on the DApp. Thanks to a reverse index built into the Registry contract, other smart contracts can also search the Registry directly. This is the foundation for the "API of APIs" functionality discussed below.

The Registry provides all the information needed to find and interact with AI services on the platform, either by listing the information in full or, when it is too long, by listing the IPFS hash. The Marketplace DApp is a convenient interface for end users, and anyone can use the information in the Registry to build similar marketplaces.

2.4 Scalable Payments with the Multi-Party Escrow and Channels

The Multi-Party Escrow smart contract ("MPE"), coupled with our atomic unidirectional payment channels, enables scalable payments in the platform by minimizing the number of on-blockchain transactions needed between clients and AI service owners. The MPE contract has two main functionalities:

- It is a simple wallet with deposit and withdraw functions.
- It is also the set of atomic unidirectional payment channels between clients and services providers.

A payment channel[15] is a tool that enables off-chain transactions between parties without the delay imposed by blockchain block formation times and without compromising the transactional security. There are several kinds of payment channels. Let us consider the simple unidirectional payment channel:

- The sender, Alice, creates an escrow contract with a given expiration date. She funds the escrow contract with the desired amount of tokens (say, 23).
- Suppose Alice wants to send 5 AGI tokens to Bob ("recipient"). Alice sends Bob a signed authorization to close the escrow channel and withdraw 5 AGI tokens from it.
- Bob checks that the authorization is correctly signed, the amount is correct, and the amount does not exceed the escrowed funds.
- Bob can close the channel at any moment by presenting a signed authorization from Alice. In this case, Bob will be sent the 5 tokens Alice authorized, and the remaining 18 tokens in escrow will go back to Alice.
- Alice can close the channel after the expiration date and take all funds back.
- Alice can extend the expiration date and add funds to the contract at any time.

In the model above, there is no way for Bob to withdraw funds without closing the channel. Otherwise, he could use Alice's signed authorization a second time and withdraw 5 more AGI tokens.

Therefore, we added a feature that allows Bob to withdraw funds from the channel without closing it, while preventing this replay attack. We used a simple, textbook solution: a nonce. We add a nonce to the message that the sender signs, and this nonce changes each time the recipient claims the channel.

With this improvement, payment channels inside MPE have the following favorable properties:

- The channel between sender and recipient can persist indefinitely. The sender can extend the expiration time and add funds to the channel. The recipient can claim the amount signed over to him at any time.
- The system is comfortably functional even when the Ethereum network is overloaded with confirmation time of several hours or even more, for the following reasons:
    - Neither the sender nor recipient needs any confirmation from the blockchain. Alice can continue to add funds, and Bob can continue to claim them in the channel, with no confirmation from the blockchain. For example, after Bob claims his funds, he can inform Alice that the nonce of the channel has changed, and she can start to send messages with the new nonce. It is easy to demonstrate that this is safe for both the sender and the recipient. There is only one condition: the

[15] http://super3.org/introduction-to-micropayment-channels/.

recipient should make sure that the transaction is mined before the expiry time of the channel.

- There is no race condition between claiming (from the recipient side) and extending/adding funds (from the sender side). The parties can use these functions at any time, and the final result will not depend on the order in which these transactions are mined.

When a user wants to call a given service, they must open a channel, add funds to it, and set an expiry date that allows sufficient time for the service to fulfill its function. Each channel is unique to a combination of client identity (sender), service identity (recipient), and daemon group identity. This allows daemons in the same group to share payment information via etdc, reducing the overall number of channels and simplifying life on the client side. Clients can be end users interacting with the platform via the Marketplace DApp or applications making calls directly or through the SDK's generated code.

2.5 The Marketplace DApp

The SingularityNET Marketplace DApp is an entry point to discovering and using AI services on SingularityNET. The DApp

- reads data from the on-chain Registry and pairs it with off-chain metadata, allowing AI services to be searched, filtered, and discovered;
- integrates the SingularityNET curation service, displaying from the Registry only those services that have been vetted and whose owners have undergone due diligence and signed legal agreements that protect user privacy and data;
- allows AI services to display custom UI components for user interactions (gathering inputs for service execution and displaying results);
- integrates with Multi-Party Escrow, enabling the user to pay for service usage;
- allows consumers to rate services they have used; this is a simple rating component that will eventually be replaced by SingularityNET's Reputation System (currently under development); and
- captures usage metrics at a consumer level.

2.5.1 Service Listing

The following diagram illustrates the various components that the DApp will integrate within different flows:

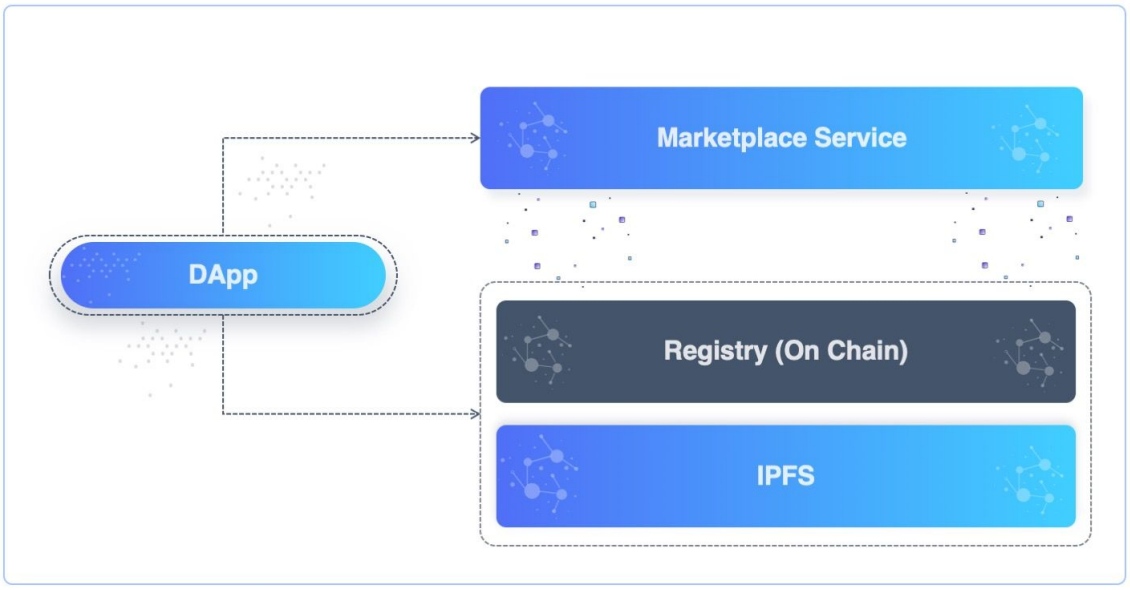

Figure 7. The various components that the DApp will integrate within different flows

- The DApp will fetch data on curated services, service tags, and votes from the marketplace service, which contains both information on which services are curated and a cache of some information kept in the smart contracts and indexed here for performance.
- It merges the above data with the agent details that it reads from the Registry and IPFS to list them.
- It will also provide users with a mechanism to upvote or downvote an agent.

The DApp includes a wallet interface, allowing users to

- deposit and withdraw funds from the escrow contract,
- deposit funds to a channel (which will go to a specific AI service chosen by the user), and
- view all open channels and the funds in each.

2.5.2 Service Execution

Once a service has been chosen, it is executed:

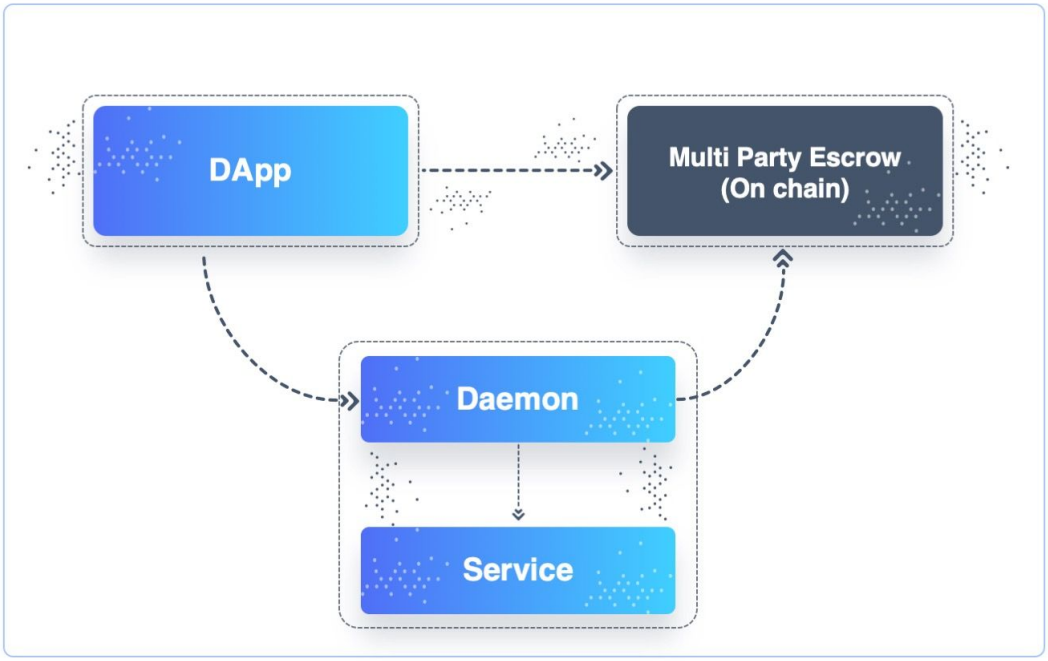

Figure 8. Service execution

1. The DApp displays the service in its interface.
2. The DApp opens a payment channel with the Multi-Party Escrow contract and ensures that there are sufficient funds to pay for the service.
3. It then invokes the service through the daemon.
4. The DApp displays the response returned by the service.

The DApp in its current version will use MetaMask to integrate with the Ethereum blockchain, which means that all transactions will be done via MetaMask.

Querying the Registry smart contract would be costly, so the DApp currently relies on a centralized, serverless component to index and search the Registry. This is merely a performance optimization, as the Registry data is still stored on the Ethereum blockchain. A future version of the DApp will remove this centralized component but will retain SingularityNET's centralized curation and display only curated services.

2.6 Developer Support Tools: CLI and SDK

2.6.1 For Service Providers: The Command Line Interface

The SingularityNET command line interface (CLI) is the primary tool for interacting with the platform's smart contracts, managing deployed services, and managing funds. It is aimed at

service providers. In the near future, it will be supplemented by a web-based dashboard and control panel.

The CLI provides commands to interface with the blockchain in the following ways:

- creating and managing identities;
- registering and managing the organizations, members, services, types, and tags on the SingularityNET Registry;
- claiming funds from customers using MPE and payment channels;
- reading and writing metadata and Protobuf specs about AI services (these are stored on IPFS, while basic service parameters can be fetched from blockchain contracts); and
- connecting to different networks like local testnets, Kovan, Ropsten, and the Ethereum mainnet.

The CLI also provides service development and deployment support. It can set up new services by generating service metadata, Protobuf specs, and code templates provided by the SingularityNET Foundation. The CLI interacts with daemons for each service.

Security-wise, the CLI follows the same guidelines as provided by Ethereum for storing the private keys. When user identities are created and registered with a client, the CLI safely stores the details on the local machine and retrieves them only when it needs to interact with the blockchain.

2.6.1.1 How the CLI Works

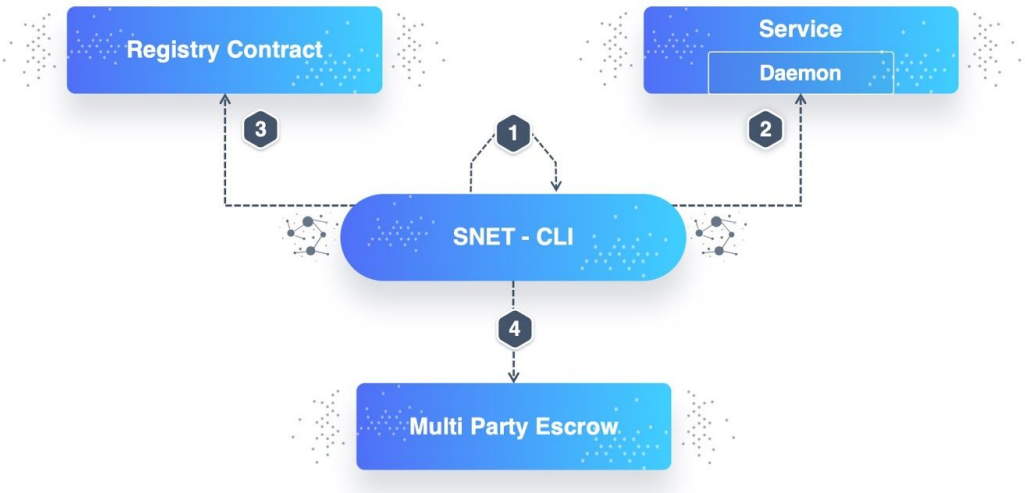

Figure 9. How the CLI works

The CLI requires and connects to four critical components:

- *User identity management*. Involves user registration, managing identities and sessions, and locking/unlocking accounts for transacting with the blockchain. This component is local to the machine where the CLI is run.
- *Daemon*. Sidecar proxy. Communicates to servers hosting AI services.
- *Registry contract*. Deals with organizations, members, services, types, and tags.
- *MPE contract*. Sends and receives funds and manages other functions related to payment channels; e.g., closing a channel or extending its expiry date

2.6.1.2 Service Registration and Deployment Workflow

Suppose an AI developer has trained a new AI that categorizes images and wants to launch it as a service on SingularityNET. The process would be along these lines:

1. Create an identity and choose the network to connect to.
2. Use the CLI to generate the basic service templates (metadata, Protobuf specs, etc.).
3. Deploy the service with the required group of daemon endpoints configured.
4. Use the identity and network from previous steps to register the organization along with required members, services, type repositories, and tags.
5. Once the service is curated, it will be shown in the Marketplace DApp. Even before that, it can be found via the Registry.
6. Channels will be created with every service used by a customer. Each channel contains the funds used by the consumer. The CLI can be used to claim those tokens from the consumer's escrowed funds.

2.6.2 For Service Consumers: Software Development Kit

The SingularityNET Software Development Kit (SDK) generates client-side libraries to seamlessly call SingularityNET services and interact with the SingularityNET platform as a whole.

While either the DApp or CLI is suitable for launching a small number of services, heavy and frequent production use of services on SingularityNET will be easier and faster through specialized client libraries generated via the SDK.

A generated client library should require only a funded wallet. It will be able to open and fund channels on behalf of the user with the multiparty escrow contract, generate well-formed requests to services, and correctly parse their responses, just like client libraries for traditional SaaS platforms.

The initial version of the SDK supports Python, currently the most popular language for machine learning and AI and a common language for glue code. In the near future, we will also provide versions in other popular languages, such as Go, Javascript, and Java.

The list of supported programming languages can be expanded at any time. Client libraries leverage the gRPC framework, so support for the programming languages targeted by gRPC "protoc" compiler is the straightforward initial step. Eventually, other languages could be supported; this would require plugins for the gRPC protoc compiler, either written by the SingularityNET Foundation or the open-source community.

As well as making service calls using gRPC, client libraries must be able to interact with Ethereum, IPFS, and other components of SingularityNET. This can be achieved with separate code for each language, or by wrapping generic libraries (for example in C or C++). For languages or operations that still are not supported, documentation will be provided to help developers integrate SingularityNET with their platforms.

A client library in a compiled language would generate calls for each service in the library and include additional helper functions to handle interactions with Ethereum, IPFS, the daemon, the MPE contract, and state channels. For example, a client library would interrogate the daemon at compile time for the encoding used by a specific service and generate method calls that would appropriately marshal requests and unmarshal responses. A client library would also store the list of channel IDs so that it does not have to rely solely on those provided by the Foundation or its partner in a transaction (which might fail, or act in an adversarial way) for such information. The developer would then integrate these libraries in their application code.

For interpreted languages like JavaScript or Python, two different options can be supported: both libraries can be generated at compile time for specific services, or more generic libraries could be used; they would download a service's protobuf specification (or load it from local storage), compile client libraries for that service at runtime (or leverage a cached, pre-compiled client), and dynamically generate service calls. The latter design is used in the DApp, CLI, and beta version of the SDK.

At the moment, the main focus of the SDK is generating client libraries and making service calls. In the future, it will be expanded to incorporate other interactions currently handled by the CLI. For example, a user should be able to use SDKs written in different programming languages to list services on the Registry; change a service's metadata; and gather and aggregate data, logs, and metrics from different daemons for different services that user controls. This would allow AI services to generate and register new services automatically, and it would enable artificial intelligence to deploy agents.

2.7 Future Improvements

The previous section described all the components of the SingularityNET platform as of February 2019, the Beta release milestone. This section describes the major conceptual innovations coming to the platform in future releases.

2.7.1 Complex Service Interactions: Service Ontology and the API of APIs

In a basic transaction on SingularityNET, a user gives tokens to a single service provider, who performs the requested AI task. However, many tasks will require a more complex combination of actions by multiple AI service providers. For example, control of humanoid robots requires multiple AI services—natural language processing, motor control, speech synthesis, etc.—to collaborate according to a particular architecture.

For a simpler example, let's say Alice requests that SingularityNET summarize a French-language website with embedded video. Her request is sent to a document summarizer service, but perhaps the top service specializes in English text summarization. Without recourse to other services, Alice's request cannot be fulfilled. However, by relying on the network of AI services, we can create an arrangement in which

1. the text on the website is sent to a document translation service, which returns an English version;

2. the embedded video is sent to a video summarization service, which returns a textual summary of key facts and events in the video; and

3. the original document summarizer service puts together these results and provides a useful summary of the website, even though it cannot understand French text or process video.

Because of these interactions between services, the document summarizer provides higher value for its customers and can earn more. Moreover, demand for the other two services grows. The result is a more vibrant marketplace. Interactions can grow more and more complex. The video summarizer can outsource face recognition, object recognition, speech detection, and speech-to-text transcription. The document summarizer may also outsource entity recognition to other services. Any of those can explicitly hire hardware services for storage or GPU access.

Out of this complex, dynamic interaction of numerous network participants carrying out complex AI services using their collective intelligence comes a SingularityNET-wide AI mind with a level of intelligence that is greater than the sum of its parts. (Notably, contemporary neuroscience's best understanding is that in the human brain, general intelligence emerges from 300 to 400 distinct subnetworks working together, each with its own architecture and set of functions and connected to specific other subnetworks in a carefully patterned way.) Furthermore, this emergent AI mind will be continually enhanced, as AI developers around the world add new nodes into the network, contributing to and profiting from SingularityNET's economy.

The platform will enable these complex interactions through three layered resources:

1. At the bottom, the type repository in the Registry allows services to state their inputs and outputs in a standard way. Service ads can say things like the following:

    a. I provide outputs of this given type ("text").

    b. I provide outputs of this given type and value (e.g., "Language" is "English").

c. I require inputs of this given type.

d. I require inputs of this given type and value (e.g., I can summarize docs if "Language" is "English").

2. Built on top of the type repository, we will have a collection of APIs, which refer to concrete type data and metadata. This allows for standard specifications for AI tasks like "face recognition," "document summarization," "genomic dataset annotation," and so forth. These APIs are a more vibrant semantic version of the standard gRPC specs already provided by the services. As the APIs are public, any developer can implement multiple APIs that provide the same service.

3. At the top level is an ontology of AI services that makes the APIs understandable and browseable. This ontology will be a directed acyclic graph with a few different roots, covering, for instance, areas of AI, application domains, and so forth. So "face recognition" would be found somewhere in the ontology, and it would be a child node of nodes such as "image processing," "deep neural networks," etc.

These three levels allow developers to find services that will accept their data as input and perform the desired function. The document summarizer AI developer in the example above needs this structure to identify auxiliary services needed to complete the job.

Provided the specifications at each level are precise enough, they also allow the emergence of AIs that connect other AIs, or programmatic service finding. These are called matchmaking agents.

AI services that serve as evaluators can be developed and launched on the network. They will specialize in assessing and rating the quality of work done by a particular service. This will allow users to search for services that offer a particular service (e.g., face recognition), are compatible with a particular API, and meet a certain standard according to an independent AI evaluator. These automated evaluations are useful for consumers, highly valuable for matchmaking agents, and a key input to the reputation system described in the next section.

Independent evaluators and public, standard APIs make it easy for new entrants to the marketplace to find customers. They can support popular APIs, enabling plug-and-play replacement of existing providers, and use independent evaluators to show the quality of their services to the marketplace.

Particularly for large enterprise customers, specialized agents can be developed to scan for new, exciting services on the market and test them on a particular problem (for example, finding patterns in a financial dataset), helping the customer select a service based on A/B testing or multiarmed bandit selection.

2.8 Reputation System

As the number of services on SingularityNET grows, there will be many AI services that perform the same function. AI service users (whether human or themselves AIs) will be faced with a

choice. The SingularityNET reputation system, by quantifying the reputation that each service has earned based on its previous work, will help them navigate this choice..

This is critical for making choices about everyday transactions in the network, and it also plays a core role in network governance and resource allocation.

Rating system design is complicated, and the SingularityNET reputation system will need to evolve along with the network. We are currently experimenting with an initial design that will combine explicit ratings by consumers, financial transaction trails, and machine learning to detect fraudulent and malicious behavior.

At the most basic level, after each exchange of services for tokens (or for other services), all parties involved are asked to rate each other on a [0, 1] scale. In this simple version, an AI service's rating is the distribution of past rating decisions. The rating can be simplified into an average value with a count showing how many times it has been evaluated. The average can incorporate some time decay so more recent ratings are weighted more heavily than those in the distant past.

Consumers and providers are not required to rate each other. Some defaults can be inferred from their behavior: if a customer withholds payment and triggers escrow arbitration, it is safe to assume they're dissatisfied with a service provider, and if a customer comes back, it can be assumed they're satisfied. How consumers and providers manage their channels can indicate trust (substantial commitments, long-lived channels) or dissatisfaction (the channel is not renewed after expiration).

Ratings can be multidimensional. This multidimensional rating system is a critical component of SingularityNET's economic and governance models. Dimensions of reputation can include general service performance, timeliness, accuracy, value for money, and so on. Other aspects reflect measures taken by the network participant to prove its good influence. The following are some examples:

- a stake deposited by a consumer or service owner, to be forfeited should its rating (in some dimension) fall below a given threshold
- a "benefit rating" component, which derives from evaluations restricted to an AI service's performance on beneficial tasks (this is key for future access to benefit tasks)
- validation by external actors, such as proof of ownership by a reputable company provided by a KYC service or a legal agreement promising to uphold data privacy regulations
- in the case of open-source software, validation via a checksum that ensures the code being advertised matches a specific release in the repository

Despite the need for multiple dimensions and conceptual aspects in a ratings system, for some purposes it is valuable to have a single-number rating—for example, to assess the basic integrity and trustworthiness of an Agent. To fulfill this requirement, the SingularityNET reputation system includes a "base reputation" rating for each Agent that is a real number between 0 and 5. For some purposes, the number 2 is used as a "base reputation threshold." For example, full participation in governance is accessible only to Agents with a base reputation of 2 or higher.

Defense against rating system frauds and attacks is a nuanced issue and will likely require a variety of machine learning models dedicated to analyzing transaction and rating patterns to detect malicious participants. This is an area of active research within SingularityNET.

2.8.1 Reputation System Concept

Since the appearance of distributed computer systems without centralized governance, verifying the reputation of participants has been a problem. This problem has been studied in its many aspects. A reliable way to determine reputation is critical for peer-to-peer marketplaces, where every node in the network can communicate with every other node.

The standard theoretical framework for such a solution comes from the Byzantine Generals Problem, which features a variable number of participants (with variable levels of trust) voting independently in order to reach a decision that is to be recorded in a public ledger so that it will be known to the entire community.

There is a risk of an attacker spinning up many malicious nodes which act together to take over the consensus in the attacker's favor. We need to design defenses against this.

Current implementations of blockchain technology use various forms of weighted voting to reach consensus. Some weight by tokens staked and others by computational power, for example. Each consensus algorithm provides certain heuristics for estimating the trustworthiness of a node.

2.8.2 Reputation System Options

Multiple inputs may be used to compute the reputation:

- First are the explicit ratings used by consumers to rate suppliers from which they have received products and services.
- Second, explicit stakes can be posed by stakeholders with respect to the suppliers they back.
- Third, there could be indirect rating information based on payments by a consumer to suppliers. For example, multiple payments may imply that a consumer values a supplier highly (a repeat customer is a satisfied customer.)
- Finally, information about the reputation of known vendors could be extracted from online news sources and social media, which in turn would need to account for their reputation.

In the simplest case, there could be just one instance of a reputation system. However, in a distributed system with a low level of trust, multiple instances of the reputation service acting together may be able to more accurately calculate reputations. These instances will cross-check each other to reach consensus on reputations. Next, multiple reputation services may compute reputations for different segments of the community in order to provide load balancing or domain-specific reputations, which can be merged by an aggregating service. Multiple types of

distributed computations may take place in the same network, so that different segments of the community have reputations computed by different groups of reputation services.

2.8.3 Reputation "Liquid Rank" Algorithm

Liquid Rank is an algorithm for computing reputation that is described in this blog post. The algorithm can take multiple parameters and inputs into account. Primarily, for different input ratings, it accounts for rating values of the ratings, financial values of the respective interactions, reputation ranks of the subjects supplying the ratings, time when the rating was provided, etc.

From one perspective, Liquid Rank can be thought of as an extension of Google's PageRank algorithm that is better suited to a marketplace, as it accounts for financial values. More expensive and more recent payments have more impact on the reputation of a supplier. As in PageRank, the greater the reputation of the rater, the higher the value of the rater's ratings.

The algorithm may be implemented either in real-time—so that every transaction changes the reputation ranks within the community—or in a stepwise fashion, where community members' reputations are calculated and updated hourly, daily, weekly, or monthly. From a practical perspective, the incremental version seems to be the most cost-effective and the specific period can be configured as a system parameter.

In extremely oversimplified form, the reputation of agent *i* at time *t* is its own reputation for previous time ($R_{i\,t-1}$) added to all the new ratings it has received from other agents *j* over the time period between *t-1* and *t*, multiplied by the reputations of these raters for the previous time, as follows.

$$R_{i\,t} = R_{i\,t-1} + \sum_j (R_{j-1\,t} * V_{i\,j\,t})$$

Reputations may be computed specifically for selected domains, such that a supplier with a high reputation in the area of its expertise might have a much lower reputation in some other domain. Within the same domain, different reputation scores may be computed for different traits—timeliness, cost, accuracy, etc. These fine-grained reputations may not be included in the first version of the reputation system.

2.8.4 Reputation Police

The goal of reputation police is to perform periodic or ad hoc inspections in order to detect malicious patterns in rating or staking activity. There are at least two problems to be solved along the way:

- *Natural cooperation versus fake cooperation*. A malicious attacker may control many agents on the network, which give each other high ratings and transfer money to each other in order to artificially inflate their reputations. We need to be able to differentiate rings of agents created maliciously from natural ones. For instance, if agent A provides agent B with image recognition service and agent B serves agent A with text summarization, and then they make payments to each other, there is no way to discern

whether the services are fake or not. We cannot discount ratings made between A and B simply because they form a loop; doing so would discourage members of the community from providing mutual services. The only way to resolve this is to flag the presence of any circle or ring of this kind and then inspect the involved agents, audit their source code, or perform an "undercover investigation," ordering services from these agents and checking that they perform valid services.

- *Temporally spanning cooperations*. Alice may pump Bob's reputation up in January, and then in July, Bob pumps up Alice's reputation. This may happen organically, or they may be collaborating to manipulate the reputation system. It should be possible to figure out if cooperation is fake or natural using the pattern recognition incorporated in our design.

Once suspicious agents are found, fraud may be confirmed with manual agent system and code inspections, automatic or manual undercover investigations, and other enforcement activities. If a suspicious agent is confirmed as fraudulent, preventive measures can be enforced in a manual, semi-automatic, or automated manner. Either the agents will be excluded from SingularityNET or their activity will be publicly reported to the community so that the providers of staking reputations can recall their stakes or highly reputable agents can be appointed to use their stakes for "corrective downvoting" against the guilty agents.

2.9 AI Infrastructure as a Service

SingularityNET is built to remove the barrier between AI innovation and real-world application. We want researchers and developers who come up with novel algorithms, techniques, and models to see SingularityNET as the best way to deploy their technology, find customers, and earn the financial and reputation rewards they deserve.

This mission requires us to handle the deployment and management aspects of AI services so AI developers can focus on what they do best and AI customers can be confident that the services they want will have high uptime, robustness, and performance and will be deployed and managed in secure, scalable environments. We will provide AI infrastructure as a service for a fee or share of revenues, similar to how app stores have simplified the mobile app economy for users and developers.

Our "AI-infrastructure-as-a-service" tools will play the role of similar tools by platforms such as AWS and Azure, but with the following design goals tailored to the needs of networked AI:

- Optimize for the computational requirements of training and deploying machine learning models. This goes beyond deep neural networks and GPU usage and considers graph processing, multi-agent systems, dynamic distributed knowledge stores, and other processing models needed to allow the emergence of networked AGI.

- Support scalable processing of stateful services, which is a challenge in current cloud platforms but necessary for many tasks such as those of conversational agents, task-oriented augmented reality, personal assistants, and others.

- Include secure support for public, private, and hybrid cloud deployments (public–private mix and edge–cloud mix).
- Dynamically optimize compute locations to maximize compute and data proximity, improving performance and reducing bandwidth costs.

We will leverage critical open-source technologies such as Kubernetes and OpenStack and support deployment of our infrastructure-as-a-service (IaaS) solution both on top of existing cloud platforms (where we make optimal use of built-in tooling) and bare metal data centers. One key consideration is using cryptocurrency mining hardware to train AI models and long-running AI reasoning and inference tasks.

2.10 Deployment in Robots and Embedded Devices

Many SingularityNET services will require powerful computing resources and will, therefore, be hosted in the cloud. However, the SingularityNET network itself is not heavyweight, and it is perfectly doable for services on the network to be embedded in robots, IoT devices, and other low-power devices. These components would run an embedded component that provides SDK and daemon functionalities, allowing them to call other services on the network while also providing services themselves—for instance, a sensor providing a data feed, which does not require much computation. This combination of edge-embedded network nodes and cloud-based computing power opens up many possibilities in IoT and robotics—some obvious, some creative and unexpected.

This will also enable robots, such as the humanoid robots from Hanson Robotics and robots from other providers, to acquire cognitive services from cloud-based SingularityNET AI services in exchange for micropayments and to receive micropayments from other SingularityNET network participants in exchange for data. It will also enable robots to carry out small economic transactions with each other based on purely local network interactions where internet connectivity is an issue.

2.11 Blockchain Agnosticism

The SingularityNET platform currently depends on the Ethereum blockchain. It may be useful or even necessary to support other existing blockchain technologies in order to broaden adoption, improve scalability, or achieve other goals. The platform architecture is designed with this possibility in mind, and it attempts to concentrate all interactions with the Registry and the MPE contract in small code components.

Throughout 2019, we will work on turning those components into libraries with as much generality as possible. A decision on which other blockchains to support, and when, has not yet been made, but encapsulating the code in libraries will ensure that as the platform evolves, the possibility of supporting other blockchains is preserved and the amount of work required will be manageable.

2.11.1 Reputation-Based Consensus

We intend to test an evolution of the proof-of-stake consensus algorithm that we call Proof of Reputation, which combines several factors: stake, activity in the network, specific rating aspects (particularly benefit rating), length of time elapsed with activity and rating levels above specific thresholds, and others. Machine learning can be used to optimize the combination of factors.

There is a significant overlap between what we intend with Proof of Reputation and the NEM blockchain's "Proof of Importance" framework,[16] so our Proof of Reputation will borrow NEM's ideas and perhaps some of their algorithms. Some component of proof-of-work may also be desirable, but we would rather solve some beneficial machine learning problem than burn cycles on cryptographic puzzles. The computational cost of these machine learning tasks varies much more than for most crypto puzzles, so this idea needs refinement over the next few years. It seems most likely that, at the end of a period of refinement and experimentation, we will end up with a Proof of Reputation framework incorporating some NEM-like aspects with a machine learning–based proof-of-work component.

2.12 Incremental Improvements to Current Components

In addition to the new components just described, we plan improvements to the existing components that will make the platform more usable and flexible for AI developers and consumers. These improvements will be released gradually throughout 2019 and include the following:

- A web dashboard and control panel for AI service owners, providing them with monitoring, metering, logging and other usage statistics for their services and allowing them to register and manage services; this is essentially a web-based complement to the CLI
- Flexible pricing models, allowing fixed-price monthly and yearly subscriptions to a service (with optional usage caps) and bulk discounts
- Asynchronous and streaming request support for AI services
- A refactored Marketplace DApp with injectable AI service UI components and no need for a centralized marketplace search service
- SDKs for Java, Javascript (Node.js), and Go
- A full-service mesh abstraction provided by the daemon that will make calls between AI services more convenient.

[16] https://nem.io/NEM_techRef.pdf.

3. Democratic Governance

SingularityNET is not only a network of AI agents but also a network of humans who use, create, rate, and otherwise interact with the AIs. Because it is a decentralized organization, the ongoing health and growth of SingularityNET will rely on democratic decision-making by network participants. A democratic process is used to make decisions regarding network operation and to allocate newly minted AGI tokens.

3.1 Reputation and Stake-based Voting

Voting is filtered by reputation; only Agents with a base reputation above the threshold of 2 will be counted. Furthermore, only Agents whose owners have been verified by appropriate KYC procedures will be permitted to vote (although other Agents can still participate in the network by offering or purchasing services).

The initial default plan is to use standard KYC methodology, likely via partnership with an external firm specializing in KYC for blockchain-based enterprises. Before year 4 of the network's operation, this will be replaced by a decentralized KYC methodology through which Agents are "KYC'd" by other Agents rather than any central authority. One possible approach is essentially a "verification federation" consisting of Agents that are democratically approved to perform KYC functions.

The amount of voting power that an owner (a verified entity that owns an agent) has regarding core network operation issues and the distribution of the future development reserve is given by the following formula:

Let stake (O) denote the total stake of owner O across all its Agents (i.e. its total amount of AGI token holdings); let stake (A) denote the stake of a particular Agent A; and let rep(A) denote the base reputation of Agent A. Let ag(O) denote the set of Agents owned by owner O.

We use the following definition:

$$\Psi(x) = \begin{cases} c * x \text{ if } x < L \\ c * (L + log_2(x - L + 2)) \text{ if } x \geq L \end{cases}$$

Here L is a boundary so that for $x < L$, the function Ψ behaves piecewise linearly and for $x \geq L$, the function Ψ behaves logarithmically (and c is just an arbitrary normalizing factor).

Then we set

$$\text{Votes} = log_2(\text{stake}(O)) * \sum_{A \in \text{ag}(O)} \Psi(\text{stake}(A))(\text{rep}(A) - 1)$$

The combination of reputation and stake in this formula gives more voting power to more highly rated entities while preventing attacks involving large numbers of sockpuppet agents that have good reputations but carry out few transactions.

At a high level, one can think of this voting formula as a sort of "Proof of Contribution":

- The use of the logarithmic function in the first term of the formula means that owners with more AGI tokens get to vote more, but that once the amount of their token ownership exceeds L, their voting power increases according to the order of magnitude of their AGI ownership rather than linearly.

- The second term in the formula means that owners whose agents are doing useful (highly reputable) things get more voting power. The use of the Ψ function is intended to avoid a dynamic in which owners are rewarded for splitting up their AI functions among many small agents, each with a tiny stake but a good reputation. For stakes up to size L, there is no reward for splitting up agents smaller than that size. For stakes above L, there is some reward for splitting up agents smaller than that size. This is analogous to a law that treats businesses smaller than a certain size differently than larger ones. The parameter L could be set initially to be equal to roughly 100,000 AGI tokens, for example.

The democratic mechanisms in the network are based on liquid (or delegative) democracy, meaning that when an agent A is qualified to vote on a decision, Agent A may also choose to delegate its voting power to some other Agent, Agent B. There may be smart contracts that allocate votes on some topics to some Agents based on metadata attached to the decisions or other more complex criteria. (For example, if you trust another Agent to do its due diligence on charitable projects, you may delegate to it your voting power for decisions about which projects are beneficial, but for no other decisions.) The network will provide standard smart contracts to automatically delegate votes, but Agents can of course use any tools they wish for this purpose.

Major changes to the network will require more votes than minor changes. By major changes, we mean, for example,

- changes in the percentage of tokens allocated to different purposes (e.g., curation rewards versus benefit tokens),

- changes to how base reputation is calculated,

- changes to the quantitative parameters governing network economics,

- any decisions regarding creating more AGI tokens beyond those initially mined, or

- key design changes like moving to different blockchains and consensus algorithms.

By minor changes, we mean things like modifications to the APIs and ontologies used in inter-agent interactions.

For decisions regarding benefit tasks, the proposed mechanism will use a combination of votes by reputable Agents and benefit votes. The network gives benefit votes to Agents in proportion to their benefit quality ratings. ("Major changes" related to benefit tasks are changes to the system that certify tasks as benefit tasks.)

3.2 Transitioning to Full Democracy

In the early phases of network development, the Foundation will make some of the governance decisions. Decision-making will transition in phases to a purely democratic governance as the network matures, with the following specifics:

- In years 1 and 2 of network operation (following the initial token issuance event), major changes are to be determined by the Foundation in accordance with the bylaws of the Foundation installed at the time of network inception, while minor changes will be determined by a 51% majority of AGI token holders.
- In years 3 and 4
    - Major changes in the operation of SingularityNET: agreement of the Foundation plus a 51% majority of AGI token holder votes
    - Minor changes in the operation of SingularityNET: a 51% majority of AGI token votes
    - Major decisions related to benefit tasks: agreement of the Foundation plus 51% of AGI token votes plus 51% of benefit votes
- From year 5 onward
    - Major changes in the operation of SingularityNET: a 65% supermajority of AGI token votes
    - Minor changes in the operation of SingularityNET: a 51% majority of AGI token votes
    - Major decisions related to benefit tasks: a 65% supermajority of AGI token votes plus 65% of benefit votes are required

3.3 Decisions regarding Benefit Tasks

In SingularityNET, a percentage of the network's assets are to be designated as "benefit tokens." (The exact percentage is determined by democratic mechanisms described in this section.) The democracy then votes on what tasks are considered "beneficial"; they would include things like

researching cures to diseases. Agents on the network can earn these designated benefit tokens by doing beneficial tasks.

A specific set of democratic mechanisms is used to decide which tasks, carried out by which Agents, are entitled to benefit tokens. As with other decision-making, this will transition from Foundation control to fully democratic control.

We introduce the role of benefit deciders: Agents authorized by the network to decide whether specific tasks fulfill the criteria needed to quality as benefit token recipients.

We propose the following:

- Each Agent gets a certain number of "benefit votes" to cast each month, based on its benefit rating.
- Benefit tasks are assigned to categories. In order for a category to be considered as a potential benefit task, it must be nominated by 2% of benefit votes cast during a month. We may create web-based tools for suggesting new tasks, soliciting votes, and easy voting.
- Once a qualified benefit decider nominates a certain task category as a potential benefit task, then the community votes on whether it should be ratified as a benefit task. Voting power on this is proportional to benefit rating. If 25% of votes cast are in the affirmative, then the task type becomes a benefit task.
- Once a benefit task is approved, any Agent capable of performing it and possessing a sufficiently high rating and benefit rating will receive benefit payment for doing it.

Research on improving the theory of benefit will initially (and perhaps ongoingly) be rated as a benefit task in order to incentivize the distributed community to contribute to this type of R&D.

4. High-Level AI Services

A large, flourishing SingularityNET will contain AI Agents of multiple types interacting in complex ways. Some AI Agents will specialize in highly abstract mathematical algorithms, others will deliver concrete end-user services and outsource their back-end algorithmics to sets of other AI agents.

The Foundation will initially seed the network with its own Agents. For this work, we make the distinction between "core AI algorithmic services" and "high-level AI services," the latter being specific concrete functionalities to end users. There may be gray areas, but this distinction adds valuable clarity beyond generically thinking about "AI Agents."

This section reviews some of the domain-specific, high-level AI services being developed by the SingularityNET Foundation's AI development team. The following section digs into the AI R&D being pursued by the SingularityNET Foundation team, some of which has already resulted in AI agents prototyped on the SingularityNET network and used within high-level AI services. Others are at an earlier stage, planned for launch on SingularityNET later in 2019 or in 2020.

4.1 Summary

4.1.1 The Need for AI Solutions

Recent advances in AI have driven an explosion of intelligent applications that will dramatically change the way we live. Figure 10 below demonstrates the vast array of enterprise companies that use AI in their product or service. Applications can be found in every vertical and functional area, from manufacturing to HR. For example, manufacturers are using deep neural networks to quickly identify manufacturing flaws, far surpassing the speed and accuracy of their existing techniques, and HR professionals are using AI to help them sift through thousands of resumes to build a short list of candidates efficiently.

In their 2018 assessment of the AI market, McKinsey Global Institute estimated that the impact of deep neural networks alone would be between $3.4 trillion and $5.7 trillion in incremental value for organizations.

All these applications are driven by AI algorithms that are packaged as AI services that provide customer solutions. SingularityNET is a network of such AI services that anyone can contribute to, making cutting-edge AI techniques available to everyone.

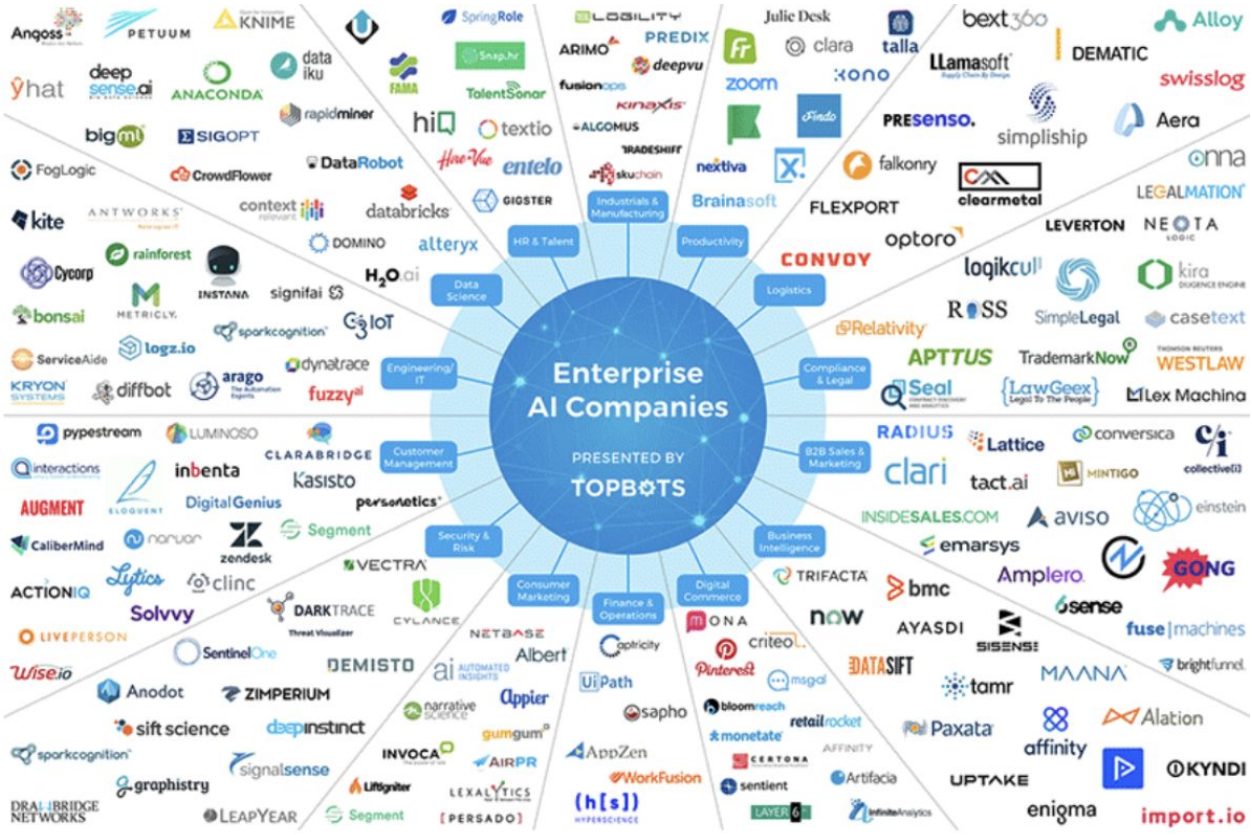

Figure 10. Companies that either provide AI as a service or incorporate AI in their product/service offering

4.1.2 What Do We Mean by AI Services?

At the most abstract level, one can think of an AI service on SingularityNET as a function with a set of inputs and outputs. The service could be a low-level service that does a specialized unit of work or a higher-level service that calls upon a series of lower-level services to complete components of its overall function.

For example, in Figure 11 below, A is the high-level AI service. It calls three lower-level services: A.1, A.2, and A.3.

E is another high-level AI service on SingularityNET. It calls on A. Other calls on the services of A are made be App 1, a software application, and S1, a smart contract on another blockchain.

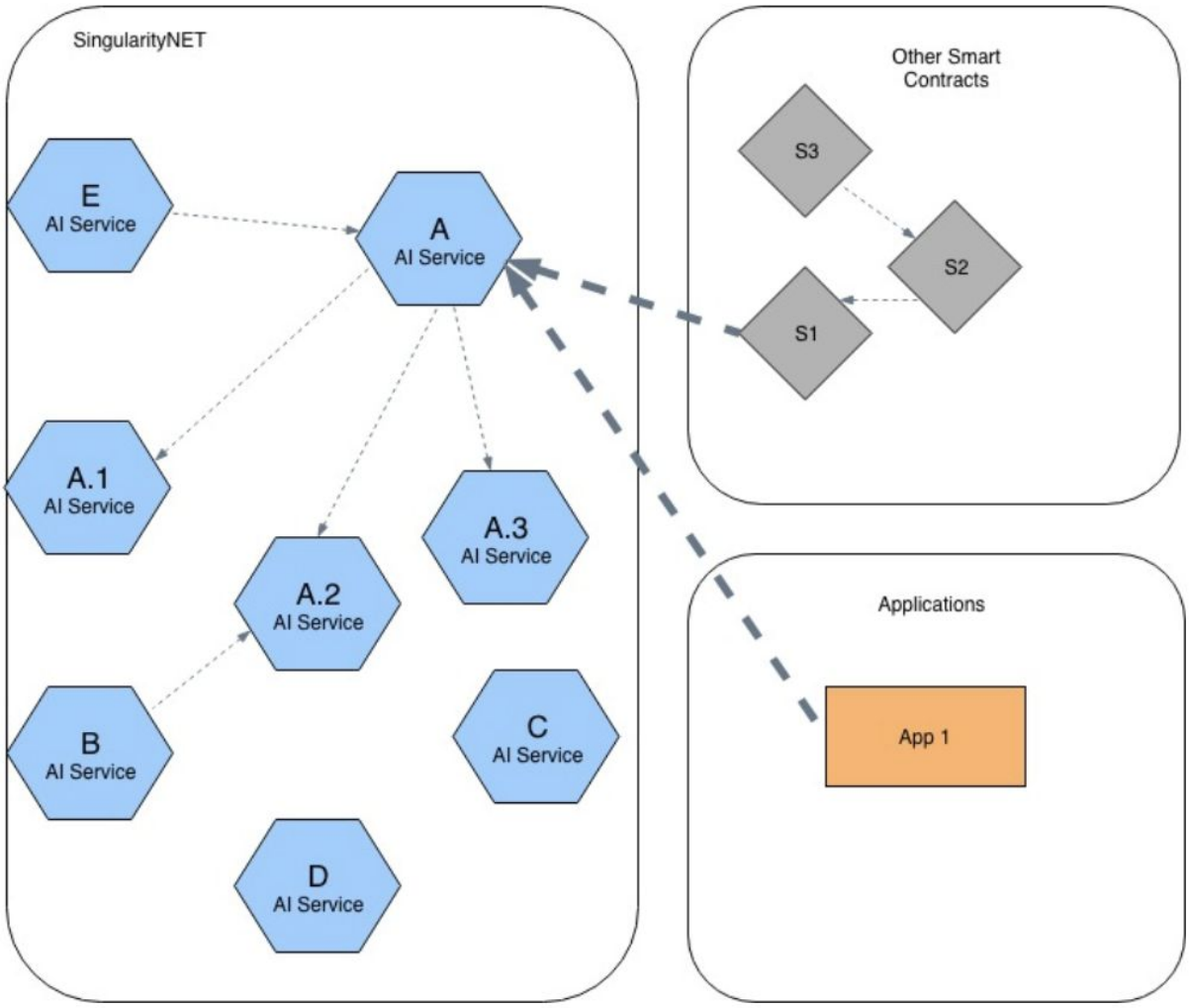

Figure 11. AI services on SingularityNET can be called in many ways: by another SingularityNET service, by a smart contract on another blockchain, by an application, or directly

An example of a high-level AI service is an image captioning service. This service would create a description of an image (e.g., "A poodle is sleeping on a kitchen floor.") The main service would be able to create a caption based on the relative positions of identified objects in the image. The service would not itself identify the objects, but would instead call lower-level services to do so and then use that information to create the caption for the image.

4.1.3 Higher-level AI Services Help Drive Growth

It is not always immediately apparent how low-level services focused on AI algorithms can be applied to solve everyday problems. This is where higher-level services help. These provide convenient interfaces to tools that solve domain-specific problems. The more of these high-level, user-facing services there are, the greater we expect the activity on SingularityNET to be.

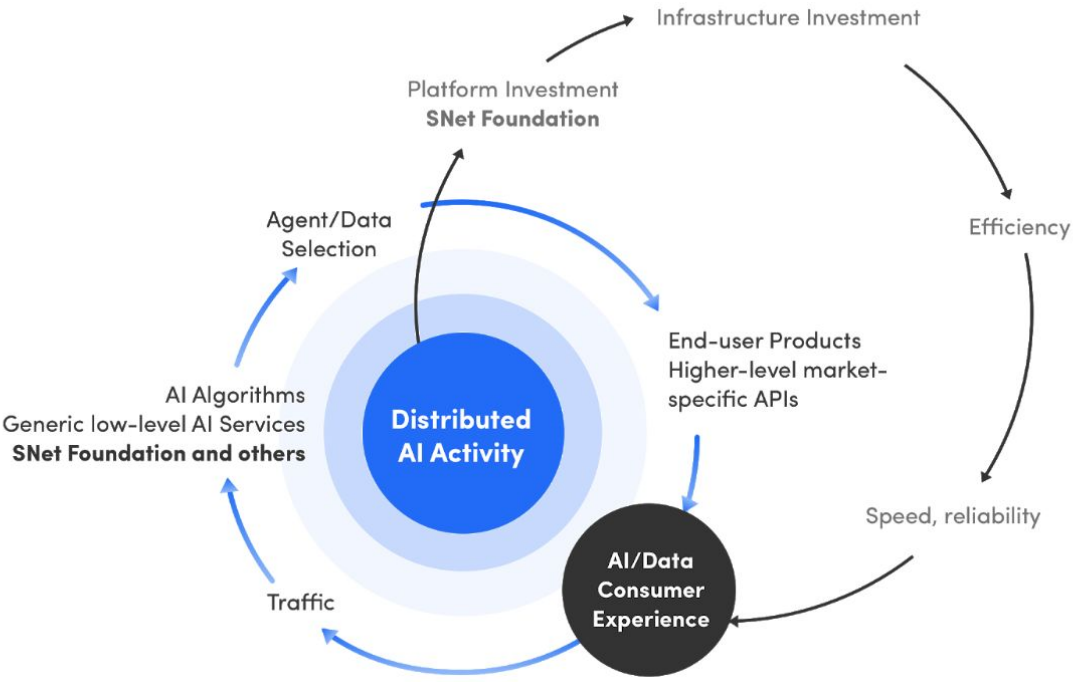

Figure 12. The SingularityNET flywheel

This relationship is illustrated in Figure 12 above, where the blue flywheel represents the activity on SingularityNET; i.e., how often services are being called. The faster the flywheel turns, the larger the blue circle will become, reflecting the greater level of activity on the network. The actions shown around the wheel drive the speed of the wheel. Starting from the top left corner, the greater the selection of AI services on the network, the wider the selection of AI projects will be – which will mean more high-level APIs will be created. This will lead to better user

experience, which in turn will drive more traffic to the network. This traffic will attract more developers to deploy AI services on the network, and so the cycle continues.

Through this activity, additional funding will be available to the SingularityNET Foundation to invest in the platform and infrastructure, which will result in improvements in speed and reliability. This, in turn, will lead to improvements in the user experience which will drive more traffic to the platform.

4.2 AI Services Provided by the SingularityNET Foundation

For our initial work on high-level AI services within the SingularityNET Foundation, we have selected four areas of focus that we describe in moderate detail below:

- Network analysis
- Social robotics
- Bio-data analytics
- Probabilistic graphical models and serious games

4.2.1 Network Analysis

4.2.1.1 Motivation

The age of big data has taught us the "network perspective": that the connections between things are often as interesting as the things themselves. Network analysis involves a broad range of tools that take the network perspective, siphoning streams of meaning from what is otherwise a firehose of information. The tools we are building on SingularityNET deal with areas like the following:

- Social network analysis and visualization, where graph algorithms from mathematics are used to describe the shape of a network as a whole and properties of the parts like centrality. This allows us to see, for example, how well network members communicate with each other, and to infer who the most influential communicators are.
- Probabilistic graphical models, where algorithms from probability and statistics are used to describe causal relationships, so we can tell, for example, the webpage that people would be most likely to want to visit or the best way to treat a patient's illness given all the facts we know about the patient.
- Network evolution, where the dynamic unfolding of network relations over time is studied using evolutionary computation and neural networks that both generate and predict network outcomes. Particular attention is paid to what causes growth and decay in networks, so we can predict, for instance, what goods and services will be in demand next year in a particularly competitive market.

- Networked artificial intelligence, the study of cooperative and competitive connections between distributed artificial intelligence programs and the processes by which these algorithms self-organize into better solutions. We use the principles of distributed AI not only to design the SingularityNET dynamics, but also as a tool to save human labor and make our AI programs serve our customers more effectively.

- Agent-based simulation of complex adaptive systems, the emulation of the virtuous and vicious feedback cycles in real-world systems to find the best policies to achieve goals. For example, we may want to explore ways to break the vicious feedback cycles of corruption in our society, develop an alert that the housing market is in a bubble, or emulate symbolic interactionist social feedback in Sophia the robot.

4.2.1.2 Examples of Applications

Our specialists in applied distributed artificial intelligence have developed various network analysis tools, each with its own practical applications. These include the following:

- A tool that reduces human labor in choosing and parametrizing AI algorithms through feedback between artificial intelligence modules. Modules are rated with tests specific to the module and with the tests set by their consumers, among others. For some classes of AI programs (for example, unsupervised algorithms such as clusterers and vector spaces), these multiple weak tests measure effectiveness better than any one strong test.

- A natural language tool to reduce the human labor of putting data into applications. This tool will interpret natural language texts (such as medical research papers) in a way that is both understandable by humans and needed by downstream AIs. It is designed to nudge unsupervised clusterers into a human-designed ontology through seeding with a few exemplars, rather than with the large lists required by supervised learning techniques, using networked relations.

- A tool to test social policy that emulates micro-level social psychological phenomena (such as cognitive dissonance and symbolic interactionism) in AI agents to explore how these micro behaviors create and react to macro social patterns. Treatment policies for social ills are applied to individual agents, where we can observe the effects on agent interactions and explore treatments. This tool has been applied to develop defense strategies against hybrid warfare campaigns that cause polarization in populations and to develop policies that alleviate corruption in societies, and has done so via award-winning analyses.

- A tool to combine the outputs of multiple disparate simulated realities into a single coherent whole using an intelligent fabric of probabilistic ontologies that automate the entry of moves in each reality and run models ahead in a gametree to evaluate the results of moves. This tool was applied to an award-winning analysis of large social systems and is useful for any data fusion application.

- A market-testing tool that incorporates adaptive economic agents. In work conducted for an insurance company, our researchers used this tool to test the effectiveness of payment innovations in increasing the quality of healthcare in America under the Affordable Care Act and to find the best pricing and offerings for new businesses in particular markets, via analyses that include higher-order effects.

- A tool to convert real-world data into a form that can be played as a game and optimized by artificial intelligence techniques. This tool was applied to personalized medicine using healthcare claims data to map out the likely effects of treatments, combining the accuracy of deep neural networks with the ability of epidemiological applications to tease out causal links in data.

4.2.1.3 SingularityNET Simulation

Alongside practical applications such as those mentioned above, we have used network-theory abstractions to design and build a miniature SingularityNET, which serves the double purpose of testing SingularityNET "policy" settings such as the reputation system and offering the same type of analysis that the full SingularityNet will offer but in a miniature form that can be run on an analyst's personal computer.

This miniature SingularityNET is a small market in which programs may send feedback to each other through price signals. Price signals serve as an assignment of credit. This simulation allows Python programs, models, and AIs to coevolve. It can be used for any coevolutionary purpose.

This simulation shows aspects of cognitive synergy between agents having emergent cognitive properties above and beyond those of the individual agents.

Because the agents in this simulation model can be made to run various AI programs, the simulation can also be made to do other things besides simulate a realistic SingularityNET. For instance, if you simulate a SingularityNET where all the AI agents are running clustering algorithms, then the simulated SingularityNET becomes essentially an emergent-level clustering meta-algorithm.

One can carry out various AI tasks (like clustering or prediction) or real world system–modeling tasks (e.g., modeling a political system or a real-world market) by the methodology of creating a simulated SingularityNET full of simulated agents running actual AI algorithms that are configured and distributed in a certain way. This approach can be used to especially good effect in situations where one AI agent's modeling process can benefit from feedback from another AI agent's modeling process.

For example, one such application is feedback between the interpretation of data and multiple overlapping disparate models of the processes that created the data—together, the data and the models create a better model as a whole. In our work on this sort of data fusion through feedback, we use specialized data processors and models that are designed to accept and adjust to feedback. These include a clusterer that can take in exemplar inputs and an agent-based model with special data-absorbing properties that integrate theory with data. A similar approach can be taken with more sophisticated AI methods, such as coevolutionary neural networks, that put parts of neural networks together with other types of AIs in a connectionist ecosystem.

4.2.1.3.1 Social Media

The algorithms of social media are suspected of contributing to many of our modern social problems and of being poor proxies for natural social interaction—but they are still essential to modern business.

Foremost on the minds of social media executives is how to preserve the quality and utility of business, social, cultural, and political interactions, but the science of how social media algorithms affect the social fabric is poorly developed.

When artificial social environments are constructed in digital space, their rules and algorithms are a proxy or stand-in for the rules that govern social interaction in the real world. SingularityNET's reputation system, for example, is an algorithmic proxy for how people determine who is authoritative and worthy of attention.

We do not assume that a painful direction of technology is inevitable, but rather seek to explore how pain could be avoided by improving the social proxy, especially to identify the qualities of natural social interactions that protect people while helping them to know each other and learn from each other.

We simulate natural social interaction using insights from social science and compare it to multiple social media and social proxy algorithms. We create measures for social values, such as democratic meritocracy and economic growth, and test them against social media social proxy algorithms.

In particular, we test popular crowdsourcing algorithms for their effect on the emerging oligarchy and explore alternatives for a way to protect democracy. SingularityNET researcher Dr. Duong's history of award-winning social science policy testing is used at SingularityNET to test our reputation system.

We want to extend these tests to include measures of SingularityNET values, such as fairness; i.e., the ability to give all software a chance to get chosen in proportion to its merit. We will exploring creating these tests by combining Kaggle-type verification with crowdsourcing, and we will explore how the reputation system should change at different stages of SingularityNET growth.

We extend the same model of oligopoly and related dynamics in social networks to tests and measures of the ability of social media algorithms to fill social needs in general, starting with SingularityNET values. In particular, we seek to demonstrate that algorithmically promoting democracy and meritocracy creates better products.

4.2.2 Social Robotics

4.2.2.1 Motivation

Our social robotics research track is focused on improving the well-being of humans through the use of natural interfaces and artificial intelligence. Instead of adapting human behavior and society to technology, this adapts technology to meet natural human behavior, creating social and cross-culturally intuitive interfaces. We aim to achieve this by researching and developing embodied humanoid robots and virtual avatars, or humanoids.

Hanson Robotics, one of SingularityNET's cofounding firms, is focused on humanoid robots capable of interacting naturally with people. SingularityNET and Hanson are designing systems to nurture multiple species of robots as next-generation interfaces for delivering AI services and applications and fostering the emergence of global artificial general intelligence.

AI drives these technologies in several key ways:

Deep learning, machine learning, and computer vision models enable auditory and visual understanding of human interactions. Accurate perception is at the root of all social interactions and shapes the quality and flow of interaction with artificial humanoids.

In this track, we emphasize perceiving social cues. We aim to advance the state of the art in multimodal emotion recognition, a field with increased visibility and relevance in recent years, and we advocate for inclusive and cross-cultural research in both data collection and modeling. We are also interested in training machines to understand relationships between people—where they are looking, at whom they are looking, to whom they are talking, the eye contact they make, and their body posture cues. While a lot of recent advancements in audiovisual perception have been made in the field of deep learning, we believe we can contribute through our holistic approach of data collection through all aspects of humans–humanoid interactions.

This track involves active research into not just perception but also the actions of our humanoids, such as speech synthesis, body gestures, and facial movements. We are developing methods of speech synthesis more emotionally expressive than the current state of the art and capable of a wide range of different intonation styles. Because of our focus on humanoid agents, we also focus on data-driven modeling of facial expression and facial expression mirroring.

We have built a dialogue and behavioral engine developed within the GHOST framework (the General Holistic Organism Scripting Tool, which described in section 5.2.6). It aims to use the OpenCog cognitive architecture to integrate our data-driven perception and expression models with the behavior of our humanoids. Our initial implementation resembles traditional rule-based approaches to dialogue; however, through the shared knowledge representation and tight integration within the AtomSpace (described in section 5.2.7) even at this first stage of design and development, we can integrate all components more tightly than traditional turn-based systems can. Over the course of research and development, we aim to replace more and more of the rule-based aspects with higher-level algorithms developed within other tracks of SingularityNET research, such as language learning, PLN, ECAN, etc.

4.2.2.2 Examples of Applications

Applications of the social robotics–related technology described above follow:

The Loving AI project was research into the emotional impact of interacting with kind, loving humanoid embodiments. We have used Sophia the robot in conjunction with GHOST (see section 5.2.7) and our emotionrRecognition deep neural network in this research.[17] We have used this configuration in IRB-approved research trials in Hong Kong in 2017 (N=26) and San Francisco in 2018 (N=35). Preliminary results show that interactions, specifically guided meditation sessions with audiovisual components, emotionally responsive dialogue, and facial expression mirroring, did lead to increased well-being and more positive feelings. The results also suggest that humanoid robots or audiovisual avatars are more effective at this than a purely audio-based interface.

The General Holistic Organism Scripting Tool (GHOST) is also being used as a conversational agent within the Mozi computational biology project. There, handcrafted rule bases have been written to guide the user through a constrained, yet natural, language dialogue. Several key components have been developed to interface this conversational agent to the experimental setup used within the project in order to provide a more fluid and natural interface to a vast plethora of possible configurations.

4.2.2.3 Plan

We have integrated the OpenCog-based GHOST tool with HEAD, the main control system for Sophia the humanoid robot.[18] This integrated system is currently being used for research and development in our offices, social robotics research trials, and some of our public events.

We are currently researching adding novel, goal-directed structures to the rule base to allow for the compositional design of skills and freeform dialogue. Also,we are improving the architecture by developing strategies for unit-testing both individual abilities and the entire architecture throughout the development process.

Our next steps are about replacing more and more of the rule-based structures with deeper cognitive understanding using our unsupervised language learning initiatives, PLN, ECAN, and other components of OpenCog and SingularityNET. We are also integrating more and more services hosted on SingularityNET in this project.

4.2.2.4 Services

Some of the specific AI services under development in the social robotics track are the following:

[17] https://arxiv.org/abs/1709.07791.
[18] https://github.com/opencog/ghost_bridge.

Action

Dialogue

- GHOST dialogue engine based on OpenCog AI

Expression

- Facial expression generation

Perception

Visual

- Faces
- Face recognition/tracking
- Face identification
- Gaze tracking
- Facial expression and emotion recognition
- Visual speaking and non-speaking detector

Bodies

- Pose tracker
- Robust person detection
- Gesture recognition
- Gait characterization

Auditory

Voice

- Speech recognition
- Voice identification
- Voice activity detection
- Laughter detection
- Multiple-speaker speech separation

- Detection of what language is being spoken

4.2.2.5 Mind-Modeling and Loving AI Development

We are currently working on extending the Loving AI pilot program to add new functions to the OpenCog AI system related to unconditional love. These functions could be manifested via any reasonably flexible robot or avatar; however, for the immediate future we will continue using Sophia for experimentation and testing. Sophia has particularly strong emotional expression capability and a global media presence, and she is a system our team is familiar with.

The work in 2019 will focus on giving OpenCog/Sophia a genuine "model of mind" for the first time. The goal will be for the AI/robot to build a working internal model of thoughts, feelings, motivations, intentions, etc. of the person with whom it is interacting. This would give a significant boost to the robot's understanding of people with whom she interacts, which we believe is required for her to genuinely express unconditional love. Of course this initial "model of mind" will not be the same as a typical human's model of other humans' minds, but it will be a start in this important direction.

One of the motivations of this work is to lay the groundwork to extend the Loving AI protocol so that the AI can learn about the person as they are interacting and to some extent bring that learning to bear in its statements and questions to the person. This will also allow for repeated sessions with the same person with continuity, as the robot builds and improves its model of the person.

This mind-modeling work integrates emotion-modeling as a result of leveraging and improving the AI framework's emotion regulation, enabling the AI's emotions to better reflect and respond to the human's emotions that it models. This will give Sophia (and any other robots or avatars controlled by the software) richer emotional expression, better emotional connection with others, and the beginnings of an understanding of human emotion in general. All of these are steps toward "skillful means" in expressing and eventually feeling unconditional love.

4.2.2.6 Social Cognition with Deep Recurrent Neural Networks

In our current social cognition research, we use the power of deep recurrent neural networks to represent social mental states, including states of cognitive dissonance, to measure and predict human reactions to information and then apply the results to improve the messages sent to persons. These results could automatically detect and alert to attempts at psychological manipulation that take advantage of human cognitive dissonance and tribalism, or they could simulate realistic social reactions on an individual level, such as feedback between Little Sophie robot and a "parent." Our scientists originally wrote similar programs using the Boltzmann machine to simulate population reactions to information operations. This tool can be applied to any case with multiple, possibly dissonant, social information messages, whether it focuses on the individual or a population.

4.2.3 Bio-data Analytics

4.2.3.1 Motivation

The explosion in the quantity and complexity of experimental data generated by biomedical research is widely recognized.[19]

> *The amount of data being produced in genomics daily is doubling every seven months, so within the next decade, genomics is looking at generating somewhere between 2 and 40 exabytes a year.*[20]

This has created a bottleneck in converting new discoveries into clinical applications[21]—the so-called "translational medicine" pipeline—and it is widely understood that machine learning and other AI approaches must be applied to increase the speed of processing data and close this gap.[22] A software infrastructure is needed to process and store the data, analyze and summarize it in an understandable form, integrate it into comprehensive predictive models of normal and pathological processes, and apply these models to diagnose and treat patients.[23]

4.2.3.2 Examples of Applications

Systematic Knowledge Discovery: Literature Aggregation and Text Mining

With an exponentially growing number of scientific publications (global scientific output doubles every nine years[24]), manual knowledge collection and curation has become an extremely challenging task. Networks of institutions continuously aggregate new knowledge in thousands of knowledge bases using both manual curation and various automated methods. A single experiment produces thousands to millions of distinct measurements that must be sifted through by referencing this existing knowledge to construct a causal hypothesis explaining the phenomena under study. Automating searches of the body of scientific literature and of the experimental findings specific to the user's research question is a crucial goal in the application of AI to biomedical research.

---

[19] https://www.cnbc.com/2015/12/10/unlocking-my-genome-was-it-worth-it.html; https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4955563/; https://www.liebertpub.com/doi/full/10.1089/big.2014.0023.

[20] https://www.washingtonpost.com/news/speaking-of-science/wp/2015/07/07/sequencing-the-genome-Creates-so-much-data-we-dont-know-what-to-do-with-it.

[21] https://www.nature.com/news/medical-genomics-gather-and-use-genetic-data-in-health-care-1.15065; https://ieeexplore.ieee.org/abstract/document/8123845.

[22] https://content.iospress.com/download/bio-medical-materials-and-engineering/bme1488; https://www.ncbi.nlm.nih.gov/pubmed/16207526.

[23] http://ieeexplore.ieee.org/document/8123845.

[24] http://blogs.nature.com/news/2014/05/global-scientific-output-doubles-every-nine-years.html.

Systematic Knowledge Discovery: Cell Population and Organ-level *In Silico* Modeling

In silico experiments and analyses use mathematical modeling and computer simulations to overcome various limitations of in vivo and in vitro methods and support the needs and research challenges of the biomedical and pharmaceutical industries. The empirical and physics-based in silico models allow preliminary discovery and testing of novel genetic and metabolic networks to be validated in experiments. The reconstruction process for genome-scale metabolic networks is well developed but labor intensive. Thiele and Palsson[25] published the best protocol in this area of research.

However, even with the impressive progress in computational biology and chemistry, the number of tissue- and organ-level simulations is limited. So far, only three organs—the mouse pancreas, the C. elegans gonad—and partial rodent brain development—have been modeled *in silico*.[26]

On the other hand, some models—for example, the human body physiology models developed within the Physiome project and the Virtual Physiological Human initiatives—have already been applied to solve some clinical problems and have brought *in silico* modeling closer to clinical translation.[27]

Diagnostic Biomarker Discovery

Biomarkers indicate alterations in one's biological state or health condition. The discovery of novel biomarkers and advances in high-throughput technologies, such as DNA microarrays and mass spectrometry, provide direct support in observational and analytic epidemiology, clinical trials, screening, diagnosis, and prognosis. Many statistical and machine learning methods have been adopted for measurement and evaluation purposes and for building predictive models based on biomedical data.

Drug Target Discovery

One of the major challenges in biomedical sciences is identifying the metabolic and regulatory pathways of disorders for rational drug design and target-oriented drug development. A simulation of a metabolic network in silico allows for simulated testing of these predicted genotype-phenotype-drug metabolic pathways.

In Silico Patient Modeling for Personal/Precision Medicine Diagnosis and Treatment Planning

The future of medicine will be highly personalized, catering holistically to each patient's unique biological blueprint. Science is beginning to uncover the unique dynamics of each person's biological structure by using machine learning tools to piece together a full atlas of an individual's genomics, proteomics, and other "-omics." Stronger models that connect our individual microbiomes to our genomes, metabolomes, and epigenomes are beginning to uncover

[25] https://onlinelibrary.wiley.com/doi/book/10.1002/9781118617151.
[26] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3896968/.
[27] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3055650/.

the delicate connections that these factors have in an individual's body. Once we fully understand these connections, we will be able to bridge accurate diagnosis techniques with highly targeted therapy (so-called theranostics), develop successful strategies for creating high-impact therapeutics, and "shift the emphasis in medicine from reaction to prevention and from disease to wellness."[28]

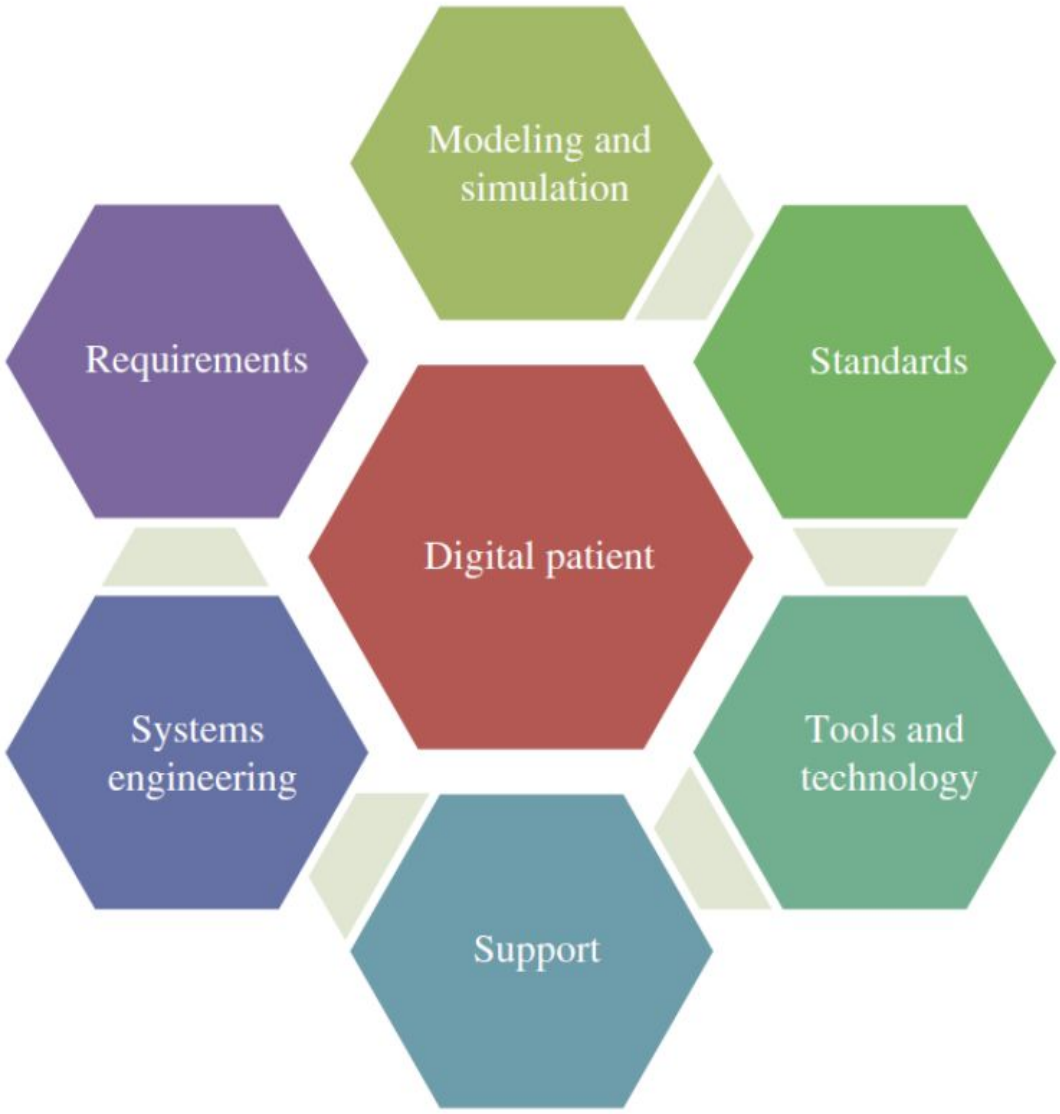

Figure 13. Main focus areas for developing and sustaining a digital patient[29]

[28] https://onlinelibrary.wiley.com/doi/book/10.1002/9781118952788.

[29] Image from *The Digital Patient*, (2016), doi:10.1002/9781118952788.

4.2.3.3 Services

The following are some specific AI services under development in the bio-data track. Some will be released with the beta version of SingularityNET in February 2019 and some are slated for later release based on ongoing work.

Supervised Classification of Binary-Valued Data Using MOSES

The SingularityNET agent accepts a data file with/and classification label information and program algorithm and validation parameters and returns a file of scored combo models and a ranked list of model features.

User Interface for Supervised Classification of SNP or Gene-Expression Data Using MOSES

A web-based interface accepts a data file input with category labels, provides an interface for setting algorithm and validation parameters, optionally solicits an Ethereum wallet address, and returns a file containing scored combo models and a ranked list of model features generated by the MOSES agent described in (1).

Annotation of MOSES Results, or other Gene Sets, Using AtomSpace Knowledge Base

A web-based interface accepts a list of gene names or reference IDs; optionally provides a screen to select from a list of reference knowledge bases, annotation types, and filtering parameters; and returns a table of the input genes and their annotations and/or a graph representation of the input genes and their annotations in a selected standard graph format.

Symbolic Regression on Genetics Datasets

The SingularityNET agent accepts a genetic data package consisting of a genetic and numerical biomarker dataset, numerical outcome values associated with each sample, and program algorithm and validation parameters. It outputs a results file containing a model that predicts the phenotype number corresponding to that genetic data package. Optionally, either FFX or MOSES algorithms can be indicated by the user.

Textual User Interface for Querying Result Sets or Knowledge Bases

A natural language query parser based on GHOST (which is described in section 5.2.6) will allow context-dependent queries, given an AtomSpace (OpenCog's database standard, described in section 5.2.7), with selected knowledge bases and analysis results as input.

Supervised Classification of Variant/SNP Dose-level Data Using MOSES

The MOSES interface will be expanded to facilitate preprocessing of whole-genome variant data and a feature-variable format indicating allelic dose.

Annotating Variant/SNP Lists or Other Genetic Base-level Data Using AtomSpace Knowledge Base

The feature annotation service will be expanded to variant/SNP lists with optional allelic dose that will be annotatable through custom AtomSpace knowledge bases and will incorporate open-source variant annotation service code at https://github.com/DEIB-GECO/GMQL and https://github.com/bulik/ldsc.

Neural Net Modeling of Sequence-Expression Links (Wrap Existing Open-Source Code)

Using the sequence/variant feature format, a sequence and tissue type is inputted and a prediction of transcript expression is made from a neural net model: https://github.com/FunctionLab/ExPecto.

Bio-NLP Textual Relationship Extraction

Using existing open-source tools to tag bioentities (small molecules, genes, proteins, cell types, organisms, diseases, etc.), OpenCog natural language processing tools will extract relations among them from arbitrary plain text or pdf documents and output an AtomSpace representation of these relationships. AtomSpace knowledge bases will be updated with new information. These knowledge bases are useful for data mining and inference processes related to user investigations.

Transfer Learning from Model Organism Knowledge Bases

One of the major challenges in genetics is to predict the functions of genes and proteins and to identify their regulatory pathways. Data mining and several machine learning techniques have been successfully applied to transfer gene annotation information between organisms.

Cell-level Hypothesis Generation from ML Results Given AtomSpace Knowledge Base and Genome-scale Cell-Metabolism Model

Given a feature list of variants, transcript expression levels, and/or protein abundances; a cell type and other context from experimental results data; and an AtomSpace containing background knowledge from public or proprietary customer sources, causal hypotheses are generated to explain the observed phenotypes associated with experimental data.

Tissue-level Hypothesis Generation from ML Results Given AtomSpace Knowledge Base and Cell Ensemble Model Including Extracellular Environment

Knowledge-base contents and inference rule bases are combined with extracellular and tissue-level context to allow us to generate meaningful hypothesis-driven inferences based on clinical and laboratory parameters of experimental sample subjects.

4.2.4 Probabilistic Graphical Models and Serious Games

Deep-reinforcement-learning methods have lately become some of the most popular algorithms in AI, but for numerous reasons they have so far not found serious application outside of a gaming environment. In our graphical model research, we are exploring ways to use networks to bring them out into practical usage; for example, to play the "healthcare game" to find the best treatment for a patient with a complicated history or to work with practically any real-world data.

The way we handle observational data is a bridge from game worlds, where we know the rules, to the real world, where we have to tease out the rules through science and epidemiological techniques.

In our graphical model research, we seek to translate real-world processes into Markov decision processes (MDPs), which represent the change in real-world states caused by different treatments. Once expressed in this form, they can be optimized by reinforcement-learning AI and other techniques. However, in order to express data in this form, attention should be paid to how to tease causal relationships out of observational data. To do this we combine epidemiological concepts (such as the "do" function of Pearl, instrumental variables, and the potential outcomes framework) with recent developments in the new accuracy of deep neural networks.

We are currently applying these methods to a curated dataset regarding the treatments of political campaigns, and intend to next use them to address healthcare data, including data from health insurance claims. However, while these are our current foci of experimentation, the scope of potential applications is extremely broad.

5. SingularityNET AI R&D Overview

For SingularityNET to achieve its goals of fostering superior AI applications across vertical markets and seeding powerful and benevolent artificial general intelligence, it must be more than just an outstanding marketplace in which narrow AI algorithms and services are matched with customers. The network must contain a certain percentage of AI agents that carry out abstract, general-purpose AI tasks. These lower-level AI agents can then be subcontracted by other AI agents carrying out more application-specific tasks and providing end-user solutions—and can learn rules more general than any one specific application area.

Toward this end, SingularityNET Foundation's AI team has been pursuing a variety of AI R&D projects, in many cases continuing and scaling up AI R&D that was already being pursued

in the open-source community or in universities. This section describes some of the most important of such efforts. The fruits of this R&D are expected to launch on SingularityNET during 2019 and 2020. As time goes on, we expect community contributions with increased intelligence and applicability, which will take most of the weight of development off the Foundation.

This body of technical and scientific effort is unique in several ways. Nowhere else on the planet, outside of a handful of large technology companies, is a comparable scale of deep AI research being conducted in a manner compatible with scalable software engineering. Moreover, while the big tech companies are focused on deep-neural-net technology that exploits their large proprietary data stores, SingularityNET's R&D is pursuing AI within a commons-based cognitive architecture.

The AI services born of this R&D work will provide direct value to sophisticated SingularityNET customers who know how to use such services directly within their software platforms, and they will provide indirect value as subcontractors to other AI agents running on the SingularityNET platform. Such AI agents may lack more advanced functionalities and would need to submit queries to the AI agents created by the SingularityNET Foundation AI team to enhance their capabilities.

5.1 Introduction

The SingularityNET Foundation AI research programme reflects the combination and intersection of multiple previously existing research initiatives, including the following:

- The OpenCog AGI project, founded by SingularityNET CEO and cofounder Dr. Ben Goertzel and colleagues in 2008 based on earlier work within the AI software firm Novamente LLC. A substantial portion of the OpenCog development team has been brought into the SingularityNET team to optimally develop and roll out OpenCog-based intelligence in a manner fully integrated with the SingularityNET platform. For general background on OpenCog, we recommend reading the CogPrime Overview Paper or the books Engineering General Intelligence Vol. 1 and Engineering General Intelligence Vol. 2.

- A related but separate research program originated in Dr. Alexey Potapov's lab at the ITMO University in St. Petersburg. It combines deep neural networks, probabilistic programming, and evolutionary learning within a common probabilistic learning-based theory of AGI. Dr. Potapov is now leading a significant team within the SingularityNET AI R&D group.

- Research on integrating perception, movement, language, emotional understanding, and expression for controlling robots and other AI characters that is being conducted by a collaboration of the AI team at Hanson Robotics and the SingularityNET team.

- Research on fusing deep neural networks with mathematical linguistics for computational language understanding and generation carried out in Sergey Shyalapin's lab in St.

Petersburg and now integrated with OpenCog-based work on probabilistic symbolic methods for language learning.

- A body of research on AI for analyzing and guiding complex systems dynamics pursued in a loose collaboration by Dr. Debbie Duong and Dr. Ben Goertzel since they worked together on government-funded research in Washington D.C. in 2002–2007, and also in a collaboration between Dr. Goertzel and the Global Brain Institute at the Free University of Brussels beginning in 2001 (and represented on the SingularityNET team by Dr. Kabir Veitas from the Global Brain Institute).

- Research on AI that integrates evolutionary learning with probabilistic reasoning and statistical learning in order to analyze biological data and other complex scientific data, that was conducted in the Hong Kong bioinformatics firm Mozi Health (now a close partner of SingularityNET).

Most of the research areas summarized here have been covered in posts on the SingularityNET research blog; the treatment here provides a more concise summary. The team hopes to provide high-quality algorithms and approaches beyond the deep neural nets that currently dominate the big tech companies' AI.

The work of the SingularityNET Foundation AI R&D team is rigorously grounded in foundational principles of AI theory and cognitive science, including the OpenCog cognitive architecture and mathematical theories of learning and reasoning. By following these underlying principles in a carefully planned way, the team can deliver AI services that provide practical value and at the same time push toward the longer-term goal of benevolent artificial general intelligence.

For simplicity, we have divided the research initiatives into two categories: (*i*)AI architectures and algorithms and (ii)measuring, modeling, and extending the SingularityNET network

However, the work being done in these two categories overlaps on both the conceptual and code levels.

5.2 AI Architectures and Algorithms

5.2.1 Symbolic Learning and Reasoning

Dr. Nil Geisweiller is leading a team carrying out advanced R&D on symbolic learning and reasoning in the OpenCog framework. The high-level motivation and conceptual background of this work is covered in research blog posts such as Introspective Reasoning Within the OpenCog Framework and Enabling Cognitive Visual Question Answering.

This work involves integrating multiple AI tools, such as the probabilistic logic networks (PLN) logic engine, the MOSES automated program learning engine, the OpenCog pattern miner, and the ECAN attention allocation system, into a common framework based on OpenCog's unified rule engine (URE).

Conceptually, the key theme is leveraging reflective meta-learning and cognitive synergy (win–win interoperation between different cognitive algorithms) to achieve higher levels of generalization and abstraction in machine learning/reasoning.

5.2.1.1 Scalable, General Probabilistic Logic

One essential initiative in this area pertains to probabilistic logical reasoning. Logical inference has been a central pursuit within the AI field since the 1960s, and modern computing resources, data sources, and theoretical advances make it feasible to integrate logical inference with probabilistic and statistical inference in an intricate manner.

The ability to relate problems (theorems) to their solutions (proofs) in a transparent manner is particularly suited to complex tasks such as bringing heterogeneous processes to inter-operate with each other, providing a link between machine understanding and human understanding, and enabling deep levels of introspection and meta-learning.

The "generalization" part of artificial general intelligence is something that logical systems are especially good at, more so than deep neural networks or other forms of AI that originate in pattern-analysis and "curve-fitting."

Although modern reasoning systems are quite sophisticated, they do have common deficiencies. They tend to be crisp (in other words, they do not handle uncertain knowledge and reasoning, or may do so in restrictive or inefficient manners) and generally inefficient, due to the inherent combinatorial explosion of building inferences.

We have designed probabilistic logic networks (PLN) in conjunction with the OpenCog framework to overcome (or at least mitigate) these deficiencies.

For instance, uncertainty is built into the logic in a mathematically rigorous way, allowing a PLN reasoner to ultimately become a substitute for both a logician and a statistician. Furthermore, by recursively applying its ability to handle uncertainty in a rigorous and general manner, PLN can express and solve problems about its own efficiency (also called "inference control" problems).

Lastly, the engine that PLN is built on top of, the unified rule engine of the OpenCog framework, has been designed with such inference control knowledge to guide its reasoning processes.

These aspects together allow for the creation of a self-improvement loop ultimately leading to more and more efficient reasoning.

The challenges in realizing this vision are significant. For instance, the transparency brought by reasoning has its computational overheads. Additionally, seeding the system with an initial efficient control policy that enables reasoning about its own efficiency is difficult in itself. Lastly, the more knowledge about inference control the system accumulates, the more costly the control decisions may become.

The OpenCog architecture addresses these challenges by providing a collection of components, often universal by nature but featuring very different sets of strengths and weaknesses, designed to be combined synergistically – a principle called Cognitive Synergy.

Some of these components, in addition to PLN, are

- MOSES, which stands for meta-optimizing semantic evolutionary search, an evolutionary program learner with some built-in capacities to learn how to search;

- Pattern miner, a frequent subgraph miner operating on the AtomSpace, OpenCog's generalized hypergraph data storage; and

- ECAN, short for economic attention networks, a resource-allocation system that dynamically estimates the importance of knowledge and processes in the system and assigns credits accordingly.

Our current research pertains to each of these components, and how to combine them for both practical goals and theoretical understanding.

5.2.1.2 Integration of Probabilistic Evolutionary Program Learning and Inference

MOSES is an evolutionary learning algorithm that extends John Koza's "genetic programming" learning framework in several important ways.

Genetic programming seeks to automatically learn computer programs by emulating the process of evolution through natural selection. In genetic programming, a population of programs is generated and evaluated on a fitness function. The unfit programs are discarded. The fittest programs survive and are combined and mutated to form a new generation. The new generation of programs then undergoes the same process of evaluation, selection, and so on.

MOSES extends this paradigm by

- considering a collection of subpopulations of programs, each focused on searching a different region of "program space";

- placing programs into a novel hierarchical "elegant normal form" which allows them to be analyzed more effectively; and

- supplementing mutation and combination with a probabilistic model of which programs will be fit and using this probabilistic model to generate new programs.

This has been shown to provide superior learning performance in a variety of cases. Applications have included genomic data analytics, financial predictions from heterogeneous data sources, and control of virtual agents in game worlds. It also yields a sophisticated framework that can require significant customization for each new application area.

Most of MOSES's computation is not explicitly framed as reasoning. This choice provides more efficiency but less flexibility. Fitness functions may be run in parallel with extreme efficiency, evaluating millions of candidates in seconds. However, a great deal of transparency is lost in the process. For instance, as only the best-candidate programs are kept for subsequent analysis, the bulk of the computation is discarded.

The key, however, is that some of MOSES's computation *is* framed as reasoning. It reasons on the probability that exploring a specific region of the search space is fruitful. These decisions may be infrequent, compared to the total volume of computation, but they are critical to the

success of the search. These anchor points constitute the bridges between efficient forms of computation (which are opaque) and the more holistic forms of computation (which are transparent), and they are the opening that allows the benefits of cognitive synergy to flow in.

Fusing MOSES with PLN and other forms of reasoning and learning has been part of the plan since MOSES was created in 2005–2007. It is expected that this fusion will allow the algorithm to scale up to learn much more complex programs than is currently possible, thus progressing toward an AGI and also enabling a great variety of additional applications.

In order to enable MOSES and PLN to work together effectively, we are now porting MOSES to the unified rule engine, with the critical decisions explicitly framed as reasoning and the rest remaining encapsulated as efficient, non-transparent computation. The existing mechanisms for inference control and meta-learning, currently present in the unified rule engine for use with PLN, will then become available to MOSES.

5.2.1.3 Pattern Mining in Logical Hypergraphs

The OpenCog pattern miner extends the existing tools for mining frequent and surprising patterns in databases, providing a uniquely powerful engine for mining frequent and surprising patterns in complex hypergraphs.

The hypergraph is the data structure used within OpenCog to represent all forms of relevant knowledge in a unified way. For an exposition of why hypergraphs are valuable as a universal AI representation framework, please read this blog post.

The pattern miner has recently been re-implemented on top of the unified rule engine for greater scalability and configurability. It shines when dealing with large amounts of data that are complexly and heterogeneously structured: natural language data, multi-omics biological data, traffic data, financial markets data, and more. In these areas, a hypergraph with logical semantics is more effective than simpler representations like relational databases or feature vectors.

One of the deepest applications of the pattern miner is to optimize AI algorithms such as PLN. It does this by looking for patterns in the choices in an AI algorithm that consistently lead to better outcomes.

For instance, given a trace of all decisions left by the unified rule engine during its execution of a run of PLN reasoning, one can apply pattern mining to understand the context, the problem to solve, the inference so far constructed, and the axioms of the system. The pattern miner then constructs inferences by applying rules and evaluates whether or not a given inference is on its way to solve the problem.

The pattern miner extracts surprisingly frequent hypergraph patterns from records of inference engine activity. One can already use these patterns to produce important inference control rules that speed up future inferences. Our recent work has shown that this can already serve as a start toward the complicated process of acquiring efficient reasoning.

5.2.1.4 Guiding Inference with Nonlinear Attention Allocation

In an AI system containing a large amount of data and/or a large number of cognitive processes, the allocation of attention becomes critical. OpenCog handles this via a system called Economic Attention Allocation (ECAN), which allocates tokens of “artificial mone”’ between the nodes

and links in its knowledge hypergraph that represent units of short-term and long-term importance to the system and its overall goals.

In collaboration with the Hanson AI team, SingularityNET has put significant effort into making the ECAN framework operate on large AtomSpaces and verifying that the way it directs attention is cognitively sensible and pragmatically effective.

ECAN has many practical uses today. It directs OpenCog's attention to allow OpenCog to generate natural language dialogue for the Sophia robot. When MOSES learns models of biological datasets and imports them into AtomSpaces, PLN can analyze them there. ECAN is essential in directing PLN's attention during this process. It will also be critical for the general guidance of the URE's rule applications.

Learning good inference control rules is very important, but even with these, controlling reasoning can be complicated because combining rules optimally takes a lot of computation. If the unified rule engine had too many control rules and had to weight every possible relevant rule to come up with the best decision, it would pause for an indefinite amount of time to deliberate, stalling the system.

Happily, we can also use reasoning to improve ECAN itself. ECAN uses a hypergraph of Hebbian links expressing how attention should be spread across data and processes, and this hypergraph is amenable to reasoning. Thus all components that can produce these Hebbian rules can be used to improve ECAN. For instance, pattern mining can be used to discover basic Hebbian rules and PLN can be used to discover finer ones, and so can MOSES.

5.2.2 Integrative Genomics as a Case Study for Integrative AI

As biology becomes an information science and information science becomes dominated by machine learning and other AI methods, it stands to reason that biology is becoming dominated by AI. To grapple with the systemic nature of disease and aging, it is necessary to do simulation modeling, data analysis, and machine reasoning regarding the multiple body subsystems across numerous datasets.

This emerging paradigm of medicine has been termed "P4 medicine that is predictive, preventive, personalized, and participatory" by systems biology godfather Leroy Hood. SingularityNET CEO and cofounder Dr. Ben Goertzel was an early practitioner of this view; since 2000 he has applied machine learning and other AI technologies to longevity and genomics, including in collaborative work with the CDC, NIH, and various universities.

In this spirit, the SingularityNET AI team has chosen biomedical data analytics—in particular the analysis of genomics data regarding longevity and age-associated diseases—as an initial testing ground for integrating multiple AI paradigms within the OpenCog framework.

MOSES is used to find patterns in genomic datasets. The small programs representing these patterns are then imported into the AtomSpace hypergraph representation. Next, the PLN logic engine is used to draw conclusions by combining the patterns with knowledge obtained from biological ontologies like the Gene Ontology project, MSigDB, etc. and with knowledge extracted from biological texts using OpenCog natural language processing technology.

For example, when applied to genomic data obtained from exceptionally long-lived people, MOSES can tell us what genes or what combinations of genes tend to have the most significant influence on these peoples' long lives. PLN and reasoning about these MOSES models, together

with other knowledge, can give us the hypotheses about *how* these genes impact aging. This can be a powerful tool for suggesting new experiments to run and for suggesting diagnostics to identify a disease state or predict future disease or longevity. It can also be applied to discover targets for either conventional drug therapy or gene therapies such as CRISPR.

In 2019, the SingularityNET bio-AI team will release a series of publications describing novel discoveries about aging and disease that have been uncovered using these methods during its 2018 research. However, these exercises in AI refinement and prototyping have importance going beyond these particular results and this particular subdomain. These methods will serve as part of the AI core of the Singularity Healthtech Studio project, and they also have general applicability beyond health-tech.

For instance, in financial services, there is a demonstrated value to applying the MOSES learning engine to combine price, volume, global macro, company accounting, and news sentiment data into combinational predictive models. Financial text analysis software is relatively mature, and an extensive amount of structured data pertaining to listed companies and their internal structures and external involvements is available. The methodology refined by the SingularityNET research team in the context of genomics AI will be adapted to play a crucial role in the Singularity Studio fintech module.

5.2.3 Neural-Symbolic Integration for Semantic Computer Vision

Neural networks have been part of the AI field since the late 1940s, but their popularity has waxed and waned over the decades. In recent years, multilayer hierarchical neural nets (better known as deep neural nets) have become extraordinarily popular due to their successes in analyzing various sorts of data, especially visual and auditory data.

A few AI researchers believe this particular tool can be refined into a universal practical AI solution and even into an architecture for artificial general intelligence. However, most AI practitioners realize that different courses require different horses. Deep neural nets are the best solution for some problems, but other problems (in particular those requiring transparent, symbolic reasoning) call for other AI techniques.

Symbolic AI approaches, such as logic engines, and program learning systems (which have been under development since the 1960s and 1980s, respectively) have historically demonstrated different strengths than neural networks. They have been better at generalization and abstraction, at planning processes (either in the physical world or in the domains of discourse and science), and at formulating novel high-level hypotheses.

For example, although computer vision tasks can theoretically be formulated as tasks of logical reasoning starting at the pixel level, such reasoning would be hopelessly inefficient. Neural nets shine when applied to computer vision tasks. By contrast, it is hard to imagine neural networks alone forming automated theorem provers.

As we aim for a more flexible, broader intelligence, the need for both symbolic and neural components becomes clearer. Ultimately, the development of artificial general intelligence will most likely require a hybrid approach, and there are almost no *purely* symbolic or *purely* emergent (subsymbolic, neural) cognitive architectures. Most architectures have elements of both, although the symbolic/subsymbolic gap is far from being fully bridged.

The field of "neural-symbolic AI" explores methodologies for combining neural network and symbolic approaches into unified AI systems that manifest the strengths of both approaches. Recent mathematical advances in AI theory using tools such as algorithmic information theory and probabilistic programming provide a coherent conceptual and formal framework in which to pursue this integration. SingularityNET AI scientist Dr. Alexey Potapov has carried out a significant body of both theoretical and practical research in this direction. Click here to see some of the relevant output of his lab at ITMO University in St. Petersburg before he joined SingularityNET in 2018.

The necessity for deep neuro-symbolic integration can be seen in the example of the image-understanding (or semantic-vision) problem. On the one hand, vision cannot be considered as a peripheral module that merely forms an input to the symbolic AI system. On the other hand, even in the vision domain, which is most favorable for deep learning, purely neural systems are insufficient to capture compositional structure and to perform reasoning (especially if transparent, interpretable results are desirable).

It should also be noted that even image classification systems can benefit from external knowledge graphs. Consider the problem of learning visual concepts and their relations: it might be necessary to both integrate neural networks with symbolic models and modify traditional neural network formalisms. Tasks such as visual question-answering (e.g., asking an AI, "What is the cat in the photo wearing?") require more top-down compositional reasoning integrated into the bottom-up image processing.

5.2.3.1 Visual Reasoning

Reasoning about visual scenes is challenging because it requires subsymbolic inductive information processing and symbolic deductive inference.

For example, suppose you want an AI to answer a question like "Are these two chairs similar?" This visual question-answering (VQA) requires top-down control of image analysis. Although this control can be implemented in the form of neural networks for simple questions using their embeddings, some VQA benchmarks have shown that this approach is insufficient and more compositional control mechanisms are required.

More-complex questions like "What size is the cylinder that is left of the brown metal thing that is left of the big sphere?" are difficult to stuff in an embedding vector of a fixed size. It is difficult to imagine that bottom-up processing can provide ready answers to such questions.

Tasks that involve visual dialogues require a sort of short-term memory. Neural models can memorize how to conduct straightforward dialogues, but for dialogues with more complex compositional structure, both symbolic inference and memory are much more suitable.

Another issue with contemporary deep neural networks (DNN) solutions is that different models are developed and trained for different tasks and even different benchmarks of the same task—such is the case for CLEVR and COCO VQA datasets.

Advancing visual reasoning has many practical applications, including video analytics, robotics, semantic image and video retrieval, augmented reality, blind-assistance systems, and more.

Due to all of these factors, the SingularityNET team approaches the problem of semantic vision and visual reasoning with the lens of cognitive architecture. Cognitive architectures are

integrative systems with working and long-term memory, knowledge representation, and reasoning engines intended for solving a wide range of tasks.

More specifically, we utilize the OpenCog cognitive architecture with its probabilistic logic network to perform deductive inference and AtomSpace to maintain the knowledge base. In the case of VQA, link grammar and RelEx2Logic modules of OpenCog are being used now to convert natural language questions to PLN queries. Neural network modules that can be executed by PLN at runtime are being developed. The primary research interest is in studying and overcoming the limitations of both OpenCog and DNNs when they are applied jointly to different visual reasoning tasks.

5.2.3.2 Concept and Representation Learning

Visual concept learning is the primary component of all semantic vision tasks that are tightly connected with representation learning. Visual concepts are learned as classifiers (discriminative models) in many models developed for solving different reasoning tasks.

These discriminative models are specialized for particular datasets, and they can be learned with enough training data. For real-world visual concepts, they are usually pretrained on labeled datasets such as ImageNet and Visual Genome. However, these datasets do not cover the whole variety of the visual world, and the tasks of unsupervised and one-shot learning are of considerable interest for general semantic vision.

Learning disentangled and semantically decomposed representations with little or no supervision is essential to solving these tasks. This is usually carried out with generative models like InfoGANs or beta-VAE. However, representations learned by generative models are much less potent than those learned with discriminative models. Getting the best of both worlds is necessary.

For instance, to learn semantic visual concepts, some degree of supervision is required. However, learning the relations between such concepts (together with both rich and disentangled representations) poses additional challenges because classes of objects are not mutually exclusive (dogs and llamas are both mammals, but they are not both pets), objects can be characterized with a variable number of attributes (a llama can be both brown and woolly), and so on.

Our research aims to solve these difficulties and to integrate corresponding generative and discriminative models in visual reasoning pipelines. This can be of help in training models with less supervision and transferring them to new datasets in not only different visual reasoning tasks but also other areas.

Suppose you want to identify a person across several images from different cameras with views that do not overlap. A model learned in one dataset will perform poorly when applied to another. In practice these models are pretrained on a labeled dataset and should be deployed onto new camera sets, for which labeling is very expensive or impossible. Combining discriminative and generative models with the decomposition of person-embedding and nuisance variables can help to mitigate this problem. The same problem—a lack of labeled datasets and difficulty of unsupervised transfer learning—is also typical of biomedical and many other applications, and the same solution should apply.

5.2.3.3 Generalization and Invariance in Deep Neural Networks

Learning visual concepts from few examples requires strong generalization. The system should identify the key features or latent variables that are invariant and separate them from variable nuisance factors. Unfortunately, generative models by themselves are not enough to solve this problem because neural networks are good at approximating functions inside the training set but not at extrapolating them beyond it.

For example, even if a decoder network is explicitly trained to reconstruct rotated images in a certain range of rotation angles, it will fail outside that range.

There is interest in achieving a general solution to the problem of generalization. Generalizing about spatial transformations can be hard-coded with the use of spatial transformers, but the more interesting problem is to achieve invariance to unknown *a priori* transformations.

Although the problem of strong generalization is external to the vision domain, we do look at the possibility of improving generalization capabilities of neural networks with more expressive formalisms.

For instance, we researched generative capsule networks (CapsNets) and hypernetworks (HyperNets), showing they performed better in generalizing certain forms of transformations. In particular, we study the possibility of learning disentangled representations with HyperNets, in which different types of factors of variation appear in the ordinary latent code and control variables.

In addition to improving learning performance in general, such extensions can have different specific applications. For example, HyperNets can be used instead of spatial transformers when the model of transformation is unknown. They can also be used to invert (i.e., to construct a decoder for) Faster R-CNN features or to design an image-matching system.

5.2.3.4 Frameworks for Neuro-symbolic Integration

Existing modular networks do combine these networks. However, they do so in a hard-coded, task-specific way where each network has an assigned task, and they include execution engines implemented in certain deep learning frameworks (such as Tensorflow or PyTorch). This makes these models automatically end-to-end differentiable, but at the cost of generality.

It seems that it would be difficult to implement the whole cognitive architecture within such frameworks, and although neuro-cognitive architectures are being developed, they are much less mature than existing hybrid architectures.

In turn, neuro-symbolic integration within hybrid architectures poses its own challenges. In particular, visual reasoning with OpenCog's PLN supposes that there is a sequence of symbolic inference steps between deep neural networks grounding visual concepts and the final answer. In order to make these networks trainable, error back-propagation through inference traces should be available, or these traces should be "compiled" into Tensorflow, PyTorch, and other deep-learning frameworks.

The SingularityNET team is looking at different possibilities to choose the most general and efficient way forward.

Furthermore, generative neuro-symbolic models might be even more challenging. Current examples of such models include variational autoencoders combined with Bayesian networks and trained with the use of a joint variational objective. However, learning the structure of the Bayesian network requires different inference algorithms.

Probabilistic programming is a general way to define generative models. However, it either uses sampling-based inference, which does not scale well to deep neural networks, or gradient descent for parameter estimation. A framework for the hybrid inference is necessary. In our work, we investigate guiding sampling in probabilistic programming by symbolic deduction and combining the sampling with gradient-based and evolutionary learning in a unified framework.

We believe deep neuro-symbolic integration is essential for scaling visual reasoning models to real-world problems.

Domain knowledge could then be incorporated into the models. For example, one can imagine a VQA or video analytics system in a boutique that uses a product catalog with categories of products (handbags, scarves, and so on) to guide the inference. Or imagine an AI tour guide of Rome that analyzes images from a user's smartphone and supports visual dialogues, making effective use of symbolic knowledge about sights.

Of course, the desirable unified framework will have many more applications not only in visual reasoning but also in natural language processing and other tasks requiring simultaneous structure identification and parameter optimization.The most basic example is learning word embeddings simultaneously with word sense induction and word clustering into categories.

5.2.4 Unsupervised Language Learning

Making AI systems richly understand human language is critical for a wide variety of practical applications and for the quest to create AGI systems that can learn from and interact with humans and comprehend our culture and values.

For most of its history, the academic field of linguistics focused on making careful formalizations of language structure (the simplest case of which is the sentence diagrams many of us learned to draw in grammar school), but since the advent of the internet there has been a greater volume of work on "statistical linguistics," the use of statistical and machine learning tools to find patterns in large volumes of textual data.

There has been a particular focus in the computational linguistics field on "supervised learning" of linguistic information, which means applying machine learning algorithms to specially prepared linguistic resources, such as collections of thousands of sentences that have been provided with sentence diagrams via the labor of human graduate students. However, the limitations of this methodology are now being recognized, and more attention is being paid to "unsupervised" methods that learn how to handle natural language by merely looking at large volumes of raw text.

There are many critical applications in this domain, but the ones we have focused on in SingularityNET, and OpenCog before, are the following:

- *Language comprehension.* Translating information conveyed in natural language into structured knowledge that can be manipulated by AI reasoning systems (enabling applications such as question-answering and knowledge-discovery)

- *Language generation.* Creating systems that allow AIs to express their internal knowledge and data in human language

- *Dialogue.* Creating systems that combine language comprehension, generation, and reasoning to carry out purposeful interactive dialogue with people

Toward these ends, we have been pursuing a project in the area of unsupervised language learning (ULL) and specifically unsupervised grammar learning: creating a software that can ingest a large body of text in a specific language, such as English or Russian, and then output a list of the grammatical rules of the language used in the text.

This software is only a part of what is needed to create powerful language comprehension, generation, and dialogue systems, but it is a critical part nevertheless.

Neither traditional linguistics nor supervised machine learning approaches have been able to comprehend the grammar of natural human languages well enough to support general-purpose natural language applications such as chat systems that can dialogue informally about general topics or scientist-assistant systems that can summarize the critical contents of research papers. Radical new advances are needed to achieve these goals, and we are well on the road to achieving them.

5.2.4.1 Approach to Unsupervised Grammar Learning

Our approach to grammar induction is novel and combines multiple algorithms and multiple AI paradigms. Early results applied using a simple version of the methodology are discussed in Bridging the Language Divide.

The critical step is to create a weighted link between each word in a sentence. In the simplest case, these weights can reflect mutual information values between the words, calculated by looking at all co-occurrences of these words across a training corpus. If words have been assigned category labels, then the weights can reflect mutual information values between categories that are calculated by looking at all co-occurrences of words in these categories across a training corpus.

Alternately, one can use a deep neural net (or other predictive language model) trained on a corpus to calculate the information value of the link between two word instances in a way that takes into account the context of the sentence and of the overall discourse or document in which the sentence occurs. There is an excellent variety of neural language modeling tools available in the recent computational linguistics literature, and our team is experimenting to see which tool provides the most reliable performance in this task.

First, the system calculates weighted links between the word pairs in a sentence. A process called maximum-weight spanning tree (MST) parsing is then used to find a graph that fulfills two conditions: it matches the sentence, and it is a valid planar graph according to Link Grammar theory. Currently, we are using an MST parser implemented in Scheme for use with the OpenCog AtomSpace, and the relevant linguistic nodes and links are represented as OpenCog Atoms.

We may create categories comprising words with similar properties, where some of the properties are calculated based on the MST parses. Examples of properties are "*being linked to the left to the word 'walk' in a lot of MST parses*" and *being linked to the left to the category C45 in a lot of MST parses*."

The results of categorization may be fed back into the link-weight–determination process to be used as input for another round of MST parsing.

Surprisingly frequent patterns may be identified in the corpus of trees obtained by MST parsing in this way. These patterns constitute the grammar rules learned by the algorithm. Some of the categories learned may be more semantic and some more purely syntactic, meaning that the rules learned can also span from purely grammatical to syntactico-semantic.

This approach can be applied to ordinary grammatical English, but it can also be applied to informal English such as tweets or text messages. Applying it to a corpus of specialized English such as biomedical research abstracts yields a specialized grammar depicting the usage of language in these sorts of texts.

For languages with complex morphology, such as Amharic, in which individual words can have multiple prefixes and suffixes and even infixes, the same algorithmic logic must be applied on the character level as well as on the word level.

5.2.4.2 Stochastic Language Generation

The above learning algorithm builds a set of Atoms in the OpenCog AtomSpace. These Atoms form a probabilistic model of the syntactic structure (and to a limited extent the semantic structure) of the input corpus. This probabilistic model may be used to generate language with the same structure.

In the most straightforward approach, sentences may be generated relatively directly from this probabilistic model. This could be considered a form of "stochastic language generation," which produces sentences that are coherent, sensible, and grammatical locally but lack a high level of semantic meaning once they go on long enough. In other words, they can generate a few sentences, but not a meaningful longer document.

Stochastic language generation can also be used to answer questions if one supplies a sentence fragment and allows the stochastic algorithm to complete the fragment. An example of such a sentence fragment is "The best thing about New York is _____."

A more exciting and sophisticated approach is to find sentences that fit the probabilistic model and also satisfy an additional semantic constraint. This generates sentences that are linguistically fluid and syntactically correct and that also represent a specified chunk of semantic content.

5.2.5 Semi-supervised Learning of Mappings from Language to Logic

Syntactic parsing of natural language sentences, as pursued in our Unsupervised Language Learning (ULL) research project, is only part of the task of translating from unstructured natural language to a structured logical representation. It gets at a certain level of semantics but fails to reach the level of more abstract semantics such as quantifiers, comparatives, multi-argument relationships, and so forth.

A related research project focuses on subsequent processing, mapping syntactic parses into sets of semantic logic expressions. This is a problem that mainstream computational linguistics has barely addressed at all, but it is critical for connecting symbolic inference engines (like OpenCog's probabilistic logic networks) to natural language data or dialogue interfaces.

During the last ten years, there have been multiple efforts within the OpenCog project to extract semantic information from natural language. This would allow OpenCog to reason on the information and build a knowledge base of the world—such as relex, relex2frame, and relex2logic. Although these efforts provided insights into the feasibility of different approaches, all of them had a common weakness: they were rule-based. These rules had to be written manually, without any automated learning.

This meant that they could handle only sentences with simple linguistic structures, were restricted to English, and couldn't scale to more complex sentences without significant effort to update multiple rules. It also meant that choosing the schema to represent additional sentence-level linguistic phenomena might break the effects of existing rules and that the rules did not account for relations across sentences.

To go beyond these limitations, and achieve robust mapping into a knowledge representation that is rich enough to represent most sentence-level linguistic phenomena, we have chosen to work with the Lojban language and associated resources.

Lojban is a constructed language with a formal grammar inspired by predicate logic. Because of its formal grammar, it provides the same syntactic unambiguity as some controlled natural languages. However, unlike them, it is not restricted to the linguistic phenomena of a root natural language and the limits of expressible semantics that come with it. This makes it possible to convey diverse day-to-day semantic constructs that exist in various natural languages.

Lojban provides a natural language with a known formal grammar (like the link grammar or induced grammar that unsupervised language learning aims to learn) and a sturdy seed for unambiguous knowledge representation. Our Lojban project aims to answer the following questions:

- How can we generate the rules that extract the semantic relations between concepts in a given sentence?
- How can these rules be used to generate natural language from semantic relations?
- How can the process of learning these rules from parallel corpora of Lojban-English pairs, involving relatively simple linguistic structures, be scaled to more complex linguistic structures?

5.2.5.1 Rule Learning

Our system aims to learn the rules that map semantics of a natural language (English for starters) to the OpenCog representation of Lojban. It does this using corpora of Lojban–English translations. The Lojban sentences will be parsed into the OpenCog AtomSpace alongside the link-grammar parse of their English translations. Then the frequent or surprising patterns in the link-grammar parses will be mined. Next, the system will mine the Lojban parses that correspond to the English sentences with frequent or surprising patterns. The patterns mined from these

Lojban parses, along with patterns for the English sentences, will form the new rules that extract the semantic information from English.

Theoretically, this approach should be able to learn rules that extract semantic information spanning multiple sentences. For example, given the input, *"I'd like to be under the sea, in an octopus's garden in the shade. We would be warm below the storm, in our little hideaway beneath the waves,"* this approach should be able to understand that the "little hideaway" in the last sentence is the "octopus's garden" in the first sentence. This requires learning corpora with samples of entire Lojban paragraphs translated into English paragraphs. Instead of looking for frequent or surprising patterns in sentence sets, we will look for patterns across sets of paragraphs.

5.2.5.2 Help Generate Natural Languages

Probabilistic logic networks, semantic vision, or other processes may generate new knowledge involving the learned rules that wasn't sourced from the natural language pipeline. If the system has to express this information to a human, then it has to express it in natural language.

To do so, the syntacto-semantic rules learned can be used to map the information into natural-language-parse representations. This, in turn, will seed a stochastic natural language generation system that will generate the sentence. The generation system will be based on formal grammar learned by unsupervised language learning or a learned probability distribution over an existing link-grammar dictionary.

5.2.5.3 Scaling to Complex Linguistic Structures

The Lojban–English parallel corpora are very limited in quantity, in diversity of topics, and in the linguistic and semantic structures expressed. However, a similar (often worse) situation exists for some natural language pairs. A recent development in unsupervised machine translation has shown promising progress in solving this problem.

This development involves word-by-word translation based on word-embedding (for learning a bilingual dictionary), a pretrained language-model for each language, and a process of recursive back-translation that involves using the model of the target language to correct the translation before the next iteration.

Taking inspiration from this approach, for the Lojban–English pair, the link-grammar dictionary and the Lojban-to-OpenCog parser will be language models with probability distributions learnt from corpora. For the last step, iterative corrective back-translation, the translations could go through a rule-learning pipeline similar to the structure of the Unsupervised-PBSMT, where the rule base plays the role of the phrase table, or runs independently of the rule-learning pipeline that will result in a corpus that will be used later for learning the rule base.

The first approach is an integrated approach; it makes it possible to include PLN or MOSES in the iterative process. The advantages and disadvantages of this approach are an area of research. The second approach is modular; it generates a parallel corpus independent of using it to learn a rule base.

5.2.6 Goal-Driven Dialogue Systems

Chatbots and sophisticated conversational AI systems have become increasingly popular in recent years, including general consumer tools like Alexa and Siri and a variety of enterprise chatbots, customer support chatbots, and so forth. Although many of these conversational AI systems are useful tools, they generally fail at being engaging conversation partners and often fail at achieving their practical purposes as well, leaving the user to achieve their goals via other means (typing, point-and-click, or talking to a human).

The technologies needed to make conversational AI agents truly useful include advances in language comprehension, speech synthesis and analysis, probabilistic common-sense reasoning, modeling of human emotions and mind states, among other things.

One critical shortcoming, however, is more foundational to the design of conversational systems: the lack of a coherent cognitive architecture governing dialogue control.

"Would you like some tea?" is a simple morsel of everyday conversation. Yet it requires complex understanding of the world: we need to know that making tea is one of the things we are capable of doing, that other humans sometimes like to drink tea, that a teakettle and tea bags are nearby, and many other things. We believe that an AI generating dialogue should have a model of the self, the other, and the situation and a desire to achieve specific goals in a particular situation relative to its model of the self and the other. If it does not, it will lack the richness and responsiveness of human conversation and will fall short in important practical respects.

Current work on AI dialogue systems involves neural networks, rule-based approaches, and hybrids, all with varying levels of functionality. However, these systems are mainly focused on modeling linguistic structures available in a particular corpus and extracting the intent from what has been said without much accounting for inputs from other sensors and without much focus on overall cognitive modeling of the context of the dialogue.

This approach may be sufficient for building a customer service assistant or a simple question-answering system that is restricted to a single domain and interacts with a single individual. Restricting the breadth and depth of the conversations makes it possible to more easily build and maintain such systems.

On the other hand, SingularityNET's conversational AI team has been primarily focused on building a dialogue system that can handle multiple domains and group conversations, take into account multiple sensory inputs and associated models, and be a delight for developers/authors to build and maintain their agents on. This work is being carried out in conjunction with the Hanson AI team at Hanson Robotics, with one key application being lifelike dialogue systems for Sophia and the other Hanson robots.

These teams are developing the general holistic organism scripting tool (GHOST), a framework based on OpenCog, as a solution to these challenges. Sophia and other Hanson robots are currently testing GHOST in social robotics.

GHOST is based on OpenPsi, a framework for modeling the relationship between contexts, actions, and the goals that are impacted by the actions and choosing an action that satisfies a prespecified utility function. As a result, GHOST can account for multiple sensory inputs to extract intents, make abstract inferences about the situation, and be driven by specified goals.

In a GHOST-based dialogue system, everything begins with high-level system goals and then subgoals that may be either explicitly specified or learned by the system. Cognitive algorithms

are then used to identify the critical features of the current context, unifying multiple inputs: linguistic inputs and any other available sensory inputs or inferences. The system then chooses whatever action is expected to maximize its goal achievement given the observed and inferred context. This may be a speech action or some other type of action, such as a movement in the case of a robot, a messaging action in the case of an online personal assistant, and so on.

In this way, the give and take of a conversation is embedded in a broader cognitive context, where goal orientation and context analysis/inference are the foundation from which the conversation flows.

5.2.6.1 Intent Classification

Extracting the intent of an utterance requires an understanding of the semantics of a sentence as a function of the broader conversation. This can be done using existing tools, to a limited extent, but the work of SingularityNET's natural language research team is expected to expand the breadth and depth of intents that can be extracted from linguistic structures.

For amplifying the signals gleaned using natural language processing, we will explore integrating auditory and visual inputs. This will require clear models of the physical and social worlds of interactions.

5.2.6.2 Dialogue Representation and State Tracking

GHOST uses ECAN, OpenCog's "economic attention allocation" framework, to determine what to give attention to in a particular context. This allows GHOST to handle multiple domains without developers or dialogue authors having to specify brittle conversational flows between the domains. This capability is the subject of current active development.

Although the current GHOST system can make transitions between prespecified domains of conversation, it is not great at tracking those transitions. How these transitions should be represented, such that they are amenable for dialogue state tracking, is also being explored currently.

Dialogue state tracking is closely related to action selection in OpenPsi. This is because the dialogue system specifies a conversational flow using OpenPsi goals, the relationships defined between them, and a utility function used during action selection.

Another avenue of current research is representing and tracking states in group conversations and determining how they evolve through time.

5.2.6.3 Authoring/Development Interface

Developing an application using GHOST requires choosing goals, modeling the relationships between them, and describing alternate conversation flows per domain. Learning the goal system is not a focus at the moment, though it will become one eventually.

We plan to investigate ways to simplify the specification of the structure of conversation flow using reinforcement learning and other machine learning approaches. Our intention is to use

GHOST-based conversational interfaces in developing (or teaching) either a domain-specific GHOST-based conversational system or a general-purpose one.

5.2.7 Distributed Processing for Semantic Hypergraphs

Many of our R&D projects use OpenCog as a foundation. OpenCog has many strengths, such as a common framework for representing knowledge used by multiple AI algorithms and a way to run multiple AI algorithms together so that they can cooperate and share intermediate representations.

However, in its current form, OpenCog also displays some operational weaknesses. For instance, it can leverage distributed processing across multiple machines only in a relatively limited way. In order to fully utilize OpenCog-based AI within the SingularityNET decentralized framework, we need to improve OpenCog's distributed scalability.

The critical task here is to redesign or re-implement OpenCog's AtomSpace hypergraph knowledge store using a distributed, robust, and scalable architecture. This will improve the ability of OpenCog to process huge volumes of data and ensure its reliability and efficiency while providing services on SingularityNET.

OpenCog uses the AtomSpace to manipulate its knowledge base and share it among its AI components, which access and update this base concurrently. So the AtomSpace can be seen as a kind of shared database that AI components query to gather information and update to improve the overall stored knowledge.

In real-world AI applications, the amount of data required to achieve interesting results tends to be very large. This means AI components need an efficient search engine to extract relevant information from the AtomSpace so that they can gather the necessary information in a reasonable amount of time. This role will be fulfilled by the Distributed AtomSpace.

The Distributed AtomSpace is a new component that will replace AtomSpace (or use it under the hood) to manipulate large OpenCog knowledge bases and provide efficient querying for AI components working concurrently, which will allow reliable OpenCog-based SingularityNET services.

5.2.7.1 Distributed AtomSpace x Distributed Database

Why are we creating the Distributed AtomSpace instead of just using an existing distributed database system? It is challenging to achieve all the scalability requirements while relying only on a distributed database management system (DBMS) because AtomSpace has at least three requirements that no DBMS covers simultaneously:

Knowledge Representation

OpenCog's knowledge representation does not fit well in relational, key-value, or even graph databases. We could map OpenCog's nodes and links representation to relational tables, key-value hashes, or graphs, but the indexing mechanisms used by the corresponding DBMS hampers the efficiency of some of the types of queries performed by AI components. An

example of a query that is challenging to optimize is pattern matching; i.e., the search for subgraphs that match a given pattern with wildcards and variables that need to be unified.

Concurrency and Data Integrity

Concurrency in OpenCog can happen on a logical level, as when components work on the same object, or on a semantic level, as when two components work on the same knowledge-base element (for example, a concept).

Data integrity policies, in this case, need to manage a configurable tradeoff between integrity and performance. In other words, data integrity policies need to balance—on the one hand—ensuring that each component see the changes the other makes to a concept and—on the other hand—allowing each component to change concept properties without delay.

Determining the value of a concept property concurrently changed by two or more components is not trivial. This operation (referred to as "merging" in OpenCog) may require complex procedures with side effects (chained changes in other elements of the knowledge base) and can differ according to the type of concept involved.

Locality of Reference

Database management systems use particular definitions of locality to implement caching and load-balance policies. Existing systems assume that locality is temporal, spatial, or one of a few other types.

None of these is a good fit for OpenCog's knowledge base. Proper cache hierarchy and load-balance policies to OpenCog must be driven by contextual locality. This type of locality tends to keep together knowledge-base elements that are semantically related in a given context.

For example, consider two AI components C1 and C2. C1 is processing texts in natural language, and C2 is analyzing spatial–temporal maps. Both of the AI components need to access knowledge-base elements that represent "numbers."

For C1, any elements representing "lexical category" that are related to a given "number" are relevant and should be kept nearby. For C2, any "lexical category" elements are entirely irrelevant. If C1 and C2 are running in the same knowledge base—for example, if both C1 and C2 are helping to control a robot—the cache hierarchy policy will need to consider different contexts to decide how elements should move in the cache hierarchy.

5.2.7.2 Extending a Distributed DBMS

The approach we are taking in our Distributed AtomSpace project is a hybrid solution that incorporates a distributed DBMS along with other custom components that mitigate the problems mentioned above. This is represented in the figure below:

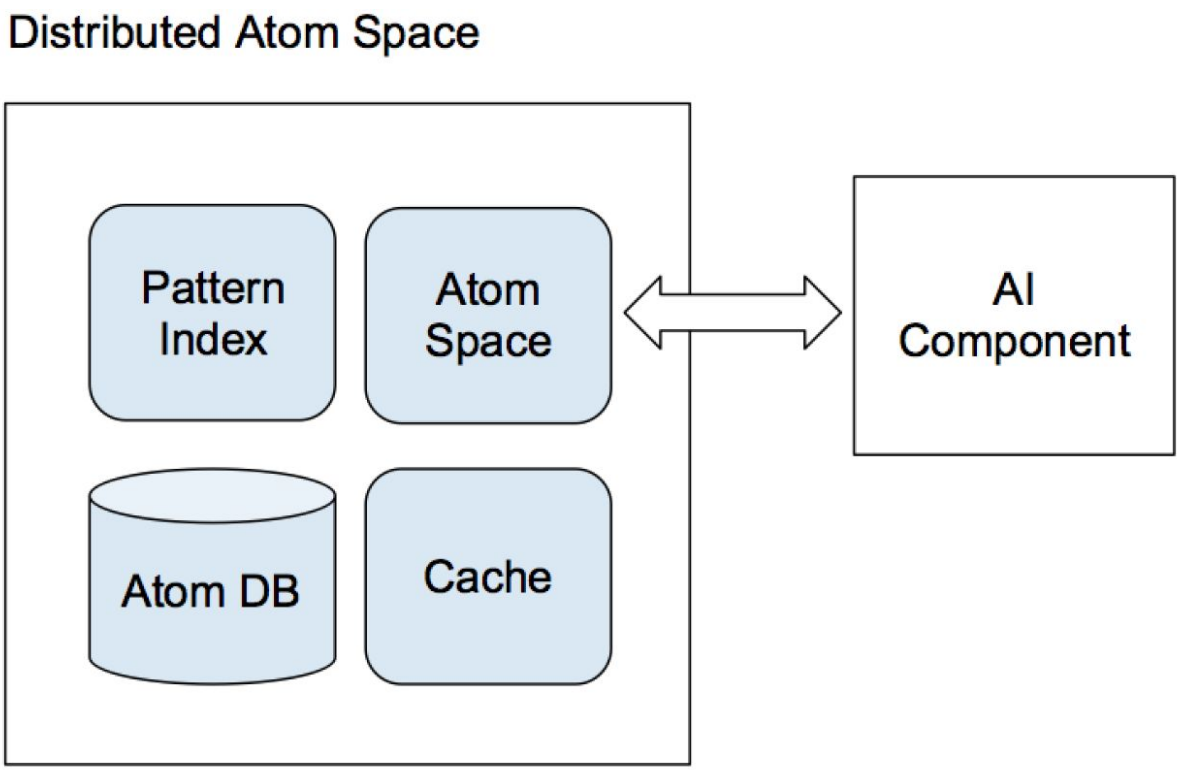

Figure 14. Components of distributed AtomSpace

AI components use the same API to access a Distributed AtomSpace as they would for a regular AtomSpace. However, a distributed DBMS (with its hierarchical cache and load balancer) is used to provide scalability.

Pattern Index is a search engine that allows queries for pattern matching. It uses an inverted index to provide efficient lookup. The index itself needs to be persisted, and this may or may not be done using the same DBMS mentioned above.

The Atom DB and the AtomSpace form the cache hierarchy. The AI Component uses patterns to define which element relations are relevant and provides this context information to drive the caching policy.

6. Measuring, Modeling, and Extending SingularityNET

SingularityNET is a platform and framework for hosting AI algorithms and systems of multiple types. Some live within individual SingularityNET agents, and others span multiple agents in a distributed way.

However, the SingularityNET decentralized AI network as a whole may also be considered a holistic AI system in its own right, in which the individual AI agents in the network are subcomponents of an overall “society and economy of minds” that is itself a kind of coherent mind.

To realize this decentralized network intelligence, we require methods for analyzing and measuring intelligence in complex, self-organizing networks. Generic AI tools such as OpenCog's pattern-mining and reasoning engines can be used for this purpose, but there is also a need for AI tools and related tools oriented explicitly toward understanding the intelligence of networks.

There is also great potential for enhancing the intelligence and efficiency of SingularityNET by making improvements to its underlying mechanisms of exchange and service discovery, among other things. However, to explore such improvements effectively requires quality tools for studying complex networks.

This aspect of R&D is expected to intensify as the network attracts a rich variety of AI agents from various authors and as the analysis and coordination of this complex network becomes a challenge that must be met by specific AI tools.

6.1 Symbolic Interaction–based Complex Network Simulation Modeling

In its early stages, the network of interactions among AI agents on SingularityNET will not be excessively complex, and analyzing and regulating the network dynamics will be straightforward. As the population of agents grows more complex, however, and the number of agents outsourcing work to other agents in complex patterns increases, then the understanding, regulation, and ongoing design of the network will become significant and fascinating challenges.

Essentially the only way to understand and manage a complex network of this nature is via simulation modeling. Toward this end, SingularityNET researcher Dr. Debbie Duong has led the development of a simulation engine capable of modeling the complex economic and cognitive dynamics that will occur within a mature SingularityNET. Experiments with this simulation framework currently focus on aspects of the SingularityNET adaptive reputation system designed by SingularityNET scientist Dr. Anton Kolonin.

The design of the SingularityNET simulation framework is based on social and economic theory and enables the use of evolutionary algorithms and neural networks within the minds of coevolving autonomous agents. Agents in the simulation form social institutions as they fulfill tasks, communicate, and employ one another in a marketplace.

The framework is designed to support micro–macro integration and model a system of social roles and institutions that emerge from social psychology and micro sociology. Lower level phenomena that are modeled include symbolic interactionism (from sociology) and cognitive dissonance (from psychology). We simultaneously model micro–macro integration in economics, including the emergence of market institutions from the utility-seeking actions of game theory.

In early experiments with the simulation, we have modeled prejudice and racism, status symbols, social class, a role-based division of labor, price, a standard of trade (money), cultures of corruption, political polarization, marketplaces, and competitor ecosystems. This preliminary work has demonstrated that the simulation framework can encompass a full range of social, economic, and cultural phenomena. This is essential, as it is difficult to foresee what sorts of dynamics will arise in a mature SingularityNET that is coupled with a rich variety of external human and computational systems.

The critical aspect in the design of our SingularityNET simulation framework is also the key component of decentralization: autonomy. This not only includes autonomy of action—the ability of the agent to fulfill its utility as it sees fit—but also autonomy of perception—the ability of the agent to freely interpret its environment, including communications, in a way that fulfills its utility.

Unlike many coevolutionary systems, in our approach, fitness for groups of agents is not averaged across the agents. Rather, agents solely seek their personal utility. Since the agents can signal to one another and have the autonomy of perception to interpret these signals, they learn to classify and contract with other agents to meet their needs.

In our simulation, the agents assign other agents to groupings that they find useful. These groupings help agents find other agents to use, and the groupings eventually become institutionalized roles when agents find it to their advantage to share their beliefs about the groupings. Agents can exert more pressure as a group than they could alone; consider, for example, the group of natural customers and the group of producers of some product type.

As in the theory of symbolic interactionism, agents find it to their advantage to behave according to the beliefs others have about them. These self-organized groupings are social roles and institutions. They emerge from micro-level utility-seeking symbolic interactionism. Thus, agents are not "assigned" to groups as in many cooperative coevolution programs; rather, groups emerge from utility-seeking interaction.

Our process of micro–macro integration and emergence improves upon the averaging techniques of cooperative coevolutionary fitness. These techniques do not attempt to assign credit (which would limit the ability of agents within groups to take on similar roles in other groups). In our system, assignment of credit and thus feedback between agents occurs through the price signal. Through the price signal, agents of different intelligence capacities are incentivized to take on the role that is most valuable to others. Agent cultures develop and last beyond the life of an individual agent. Old AIs guide new AIs into their most lucrative role in this self-organized system.

This general simulation approach has been leveraged in different forms in Dr. Duong's previous work and has been labeled symbolic interactionist simulation of trade and emergent roles (SISTER). In our early SingularityNET research we have used SISTER to model real societies (to test social policy) and to generate AI systems in artificial societies.

6.1.1 Agent Selection in Simulated Networks

In our current work, we apply SISTER to help diverse AIs of different capacities self-organize to solve problems. These AIs cooperate and compete in more complex ways than present-day AI ensembles.

SISTER creates a semantic space by which agents can learn their role in an emergent system. In our simulation of SingularityNET, many different AIs come together to solve the problem of choosing, parameterizing, and assembling Python programs—and these Python programs are themselves AIs.

We are currently using CMA-ES as an AI in each SISTER agent, but any scalar or discrete AI method can be used in any combination of agents. There is a blackboard (that can be either global or localized) on which agents bid for other agents. Customers develop tests based on their

previous experiences of which suppliers provide good services and use these tests to choose suppliers. Customers also learn criteria regarding the supplier's price. Lastly, there are criteria regarding the "sign" the supplier displays, as discussed in the next section.

6.1.1.1 AI Agent Signs in SISTER Systems

The sign can be a float vector or a string of zeros and ones. The sign starts out meaning nothing, but agents learn to use it to identify themselves as they become more successful at making their customers succeed. In doing so, the sign comes to mean an implicit set of requirements.

The process by which signs come to have meaning and agents group into types that can solve use case is as follows:

In the beginning of the simulation (before agents have learned anything), a customer (Carl) is looking for a "clusterer" service to cluster a dataset. Carl requires that the clusterer pass a silhouette test and will take it for any price under 23 AGI tokens. Because he hasn't yet learned what sign to look for, he generates a random requirement for the sign.

Two clusterers on the network pass Carl's test and have a price under 23 AGI. One of them (call it Chloë) happens to have a sign closer to the arbitrary sign Carl seeks. Carl uses Chloë's services, and both improve their utility on the network. Thus, Carl learns (by reinforcement) to look for a similar sign next time he wants to hire a clusterer. Chloë The Clusterer learns to display such a sign for its clusterer services. Since the customer is looking for a sign, other agents displaying the sign can come in on the trade. In this way, the sign comes to mean "clusterer."

Let us now suppose that Chloë subcontracts with another service, a vector-space creator named Victor. In this case, Victor will get money only if someone hires Chloë. So the sign that the vector-space creator displays to its clusterer customer will come to mean "I can help you pass the silhouette test."

However, Chloë is not the only customer that Victor The Vector-Space Creator has. Victor also sells services to another clusterer, who is named Claude. Claude is not hired on the basis of the silhouette test; he is hired by Molly, who has been tasked with making a model that fits a particular dataset well.

So Victor's sign now also means that he can help Molly fit her models to the data by means of helping Claude The Clusterer. Victor's sign means he can help in a particular, emergent set of use cases that provide utility, given the capabilities of the competition.

Now suppose a new agent comes into the simulation and displays a sign indicating it can provide "clusterer" services. It will receive requests to fulfill myriad requirements, and it will receive offers from vector-space creators because the network has learned that clusterers require the services of vector-space creators. If the agent can perform the services effectively, it will receive selective pressure to keep displaying the sign. If it can do a better job than its predecessors, it may not buy the services of vector-space creators but rather something else to please its customers. If it cannot do as well as its predecessors, it receives selective pressure to display another sign denoting something at which it is more competent.

In SISTER systems, the signs form a utility space of similar requirements and their typical implementations. As the simulation goes on, agents become competent, a competitive system develops, testing thresholds rise, more tests are required, price thresholds rise, and signs come to

have more precise meanings. The tests for unsupervised methods (like clusterers) are weak, but the aggregated effect of multiple weak tests, from groups of customers, groups of their customers, and so on, is effective in shaping performance.

6.1.2 Social Algorithms to Solve Social Problems

In prior work that involved applying SISTER for US government agencies, we used coevolutionary AI to help solve social problems. One line of our research is to understand the origins of prejudice and stereotyping and how policy measures can mitigate the problem.

Using agents that each have a recurrent neural network for a mind, we simulate how persons of different ethnicities or genders can, though they have equal inner talent, nevertheless come to be misunderstood institutionally and be forced into social roles that do not fulfill their talent potential.

We also study social media algorithms such as PageRank that overrate persons of one stereotyped class and underrate persons of another stereotyped class. We look at how this forms social class and oligarchy and at alternative social media algorithms that are not only more just but result in better utility for almost every stakeholder.

We then analyze digital protections against prejudice and oligarchy. We look at natural social protections that are absent in social media and that result in vulnerability to psychological manipulation. In particular, we study information warfare and its effect on populations through models of psychological cognitive dissonance implemented with recurrent neural networks in agent minds. We show the development of political polarization in society as a whole and compare it to more standard political models, such as those based on median voter theorem when not under the influence of information warfare.

Given this model, we test policies to heal the divide. We look at the soft equilibria of social institutions and how corruption can spread through society via hybrid warfare tactics that undermine social institutions. We examine the effects of such tactics on individuals as well as how society may heal from the resulting breakdown.

It is important to note that many of the types of phenomena that we have modeled in actual human societies can be expected to occur in a mature SingularityNET network. The formation of oligarchies, the overrating of classes of agents due to stereotypes, and the emergence of corruption and polarization are all phenomena that can occur in AI agent systems as well as in human societies.

In order to have an adequately functioning large-scale SingularityNET, we need to prevent these sorts of phenomena achieving any level of prevalence; the best way to do that is to understand the conditions under which they can arise and the potential methodologies for counteracting them by experimenting with appropriate simulation models.

To apply this sort of modeling in a fine-grained way to real-world situations (among either humans or AI agents) requires using real-world data to condition critical aspects of the simulation. We offer three ways to get the social environment into a model.

One is through creating a Markov decision process in which the effects of policies are taken from observational data in a manner that teases out causes. We use instrumental variables, the “do” from Pearl, and the potential outcomes framework to ensure that the causal connections of the Markov decision process are entered correctly.

Next, we recognize that in a complex system, we can't just put the data in directly; feedback, network effects, and mutual causation have created the observational data. We use coevolutionary tools and run a simulated form of the data through multiple models to recreate the same set of virtuous and vicious cycles found in the observational data records.

Finally, it may be difficult to get the data into the ontology of the simulation (particularly for free-text data). We offer a clustering technique that is based in coevolution in which only a few exemplars are needed to seed the clusterer to the same ontology as the model.

6.2 Tononi Phi for Measuring Integrated Information in Complex Cognitive Networks

It is difficult to measure and analyze the overall state of complex cognitive AI systems. We are currently experimenting with the Tononi Phi coefficient as a tool for measuring the overall level of "integrated information" in various complex AI networks, including SingularityNET itself, the OpenCog AI framework, and Hanson Robotics' humanoid robot Sophia.

In 2004, University of Wisconsin psychiatrist and neuroscientist Giulio Tononi created a detailed and evolving system and calculus for studying and quantifying consciousness that he called integrated information theory (IIT). The Phi coefficient defined in this framework measures the level of holistic information integration in a system. It has been posited by Giulio Tononi and others that Phi is a fundamental measure of the "level of consciousness." Regardless of its interpretation, it is an interesting measure of a consciousness-related property of a complex cognitive system. Targeting Phi maximization may lead to worthwhile results.

Phi also has potential as a feedback mechanism to tune parameters of complex dynamical systems and encourage the emergence of high-level network structures. In this manner, IIT and Phi may serve as additional tools for analyzing complex network structures, developing true AI-based social and emotional robotic systems, and creating better and more intelligent general AI services.

We have experimented with measuring Phi across time series generated by OpenCog's ECAN attention-allocation module while the system parsed and semantically analyzed a series of short documents. We have also calculated Phi values while the OpenCog system controlled the Sophia humanoid robot as she led a person through a structured meditation session.

In the latter experiment, due to the difficulties caused by computational growth, we also experimented with a new methodology: pre-preprocessing the data using independent component analysis (ICA) to reduce the problem dimensionality. In both experiments, we compared the Phi value time series obtained with the time series of events in the external situation and behavior of the OpenCog system. Qualitatively, we found correspondences between changes in Phi and changes in the situation and behavior of the cognitive system, which provides preliminary validation for the methodology.

As we continue IIT and Phi experimentation, one of our goals is to create an additional tool to measure complex emergent phenomena in order to guide the development of better-performing, more-intelligent AI services. By providing a robust theoretical method of quantifying interconnectedness, IIT can ultimately lead to improved cooperation both within and among various system components.

6.2.1 Tuning Parameters

As a measure of system connectedness, Phi/IIT will be used to improve many of the complex network structures underlying many AI services. IIT and Phi measurements can contribute to better-quality services by establishing an additional feedback loop to tune the many free system parameters in complex dynamical systems. The diagram below illustrates a representative feedback loop in the ECAN module. PLN, MOSES, and other tools would have similar diagrams.

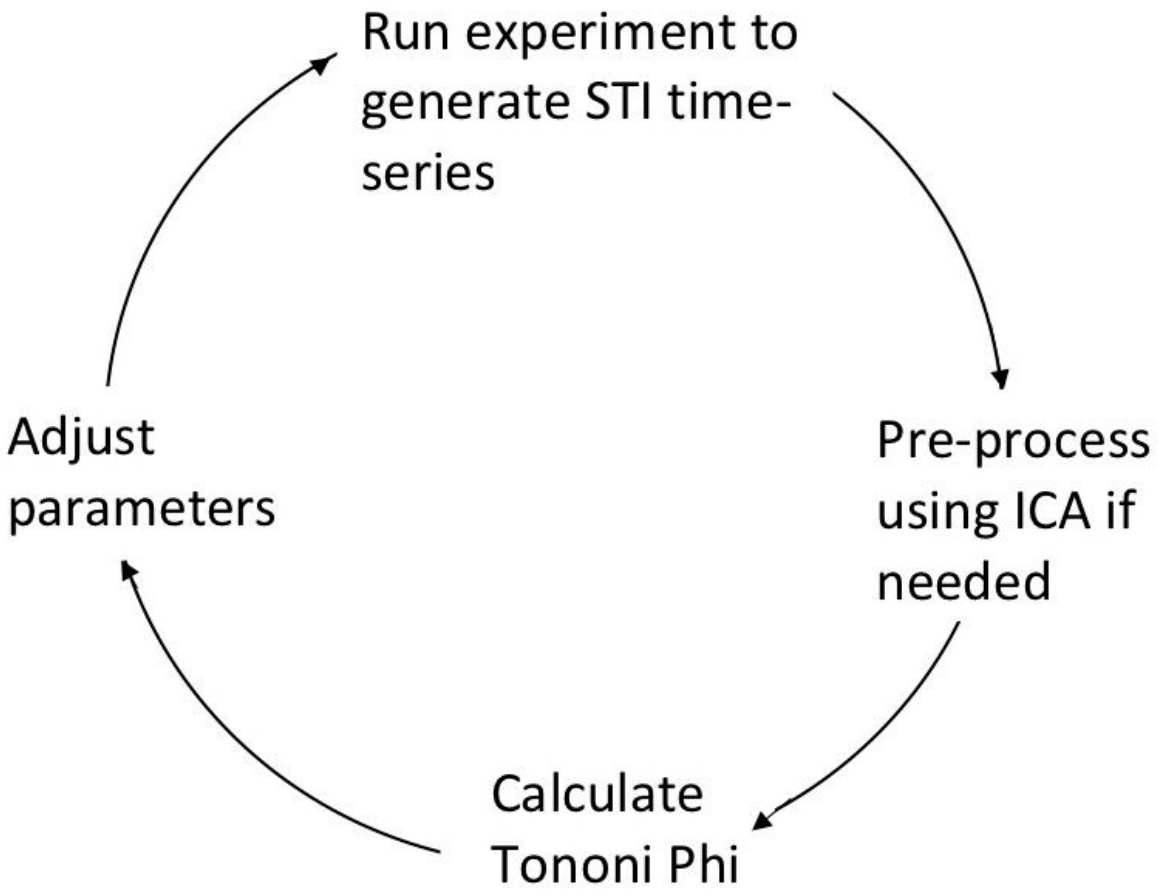

Figure 15. A representative feedback loop diagram for the ECAN module

6.2.1.1 Quantifying Cognitive Measures in AI Systems

The concept of cognitive synergy is central to the OpenCog AI framework. We will apply IIT and Phi measurements to larger-scale structures to facilitate both inter-module and intra-module parameter tuning. Exploring IIT in this manner will create synergies between a number of

OpenCog's primary modules and enable approaches to general problem-solving that are more comprehensive and fundamentally more robust (i.e., better able to overcome "stuckness"). OpenCog modules and tools with which we will experiment include the following:

- *PLN.* A probabilistic logic engine based on forward-chaining and backward-chaining within the probabilistic logic networks formalism
- *MOSES.* An evolutionary program-learning framework incorporating rule-based program normalization, probabilistic modeling, and other advanced features
- *ECAN.* "Economic attention allocation" engine based on nonlinear dynamics that assigns attention according to the spreading of ShortTermImportance and LongTermImportance values and Hebbian learning
- *Pattern mining.* Greedy hypergraph pattern mining based on information theory
- *Clustering and concept-blending* Heuristics for forming new ConceptNodes from existing ones.

Higher Phi values should correlate with interesting cognitive behaviors emerging from such systems (and initial experimentation suggests that this is the case). Phi measurements of "system connectedness" can enhance services by, for example, quantifying the synergy between different network substructures, modules, and tools.

6.2.2 Impacts on Specific SingularityNET Services

In social and emotional robotics, preliminary demonstrations suggest that Phi correlates with qualitatively interesting behavior during Sophia's meditation sessions. We plan to pursue further research into this relationship between Phi and Sophia's perceptions and actions within the Loving AI project, a collaboration with Hanson Robotics, iConscious, and other parties focused on creating robots and avatars that display unconditional love toward people. Through Phi measurements during human–robotic interaction, Phi research could also improve facial expressions and movements as well as language understanding and speech synthesis.

The ability to improve system performance and intelligence (both within individual components and via synergies between components) should lead to better SingularityNET services in multiple domains, including our initial set of network analytics, social robotics, and bio-data analytics.

Specifically, in the area of network analytics, in addition to improving parameter tuning, overall Phi measurements can

- provide improved measures of causal relationships for social network analysis and visualization and probabilistic graphical modeling as a result of Phi's inherent cause-and-effect repertoires;

- help quantify the degree to which system phenomena emerge from simple rules;
- improve the study of cooperative and competitive connections in networked artificial intelligence between distributed AI programs and the processes by which these algorithms self-organize into better solutions; and
- aid in the study of virtuous and vicious feedback cycles in real-world systems by quantifying a system's integrated connectedness and improving its cognitive synergies, ultimately helping find the best policies to achieve goals.

6.3 Offer Networks for Optimizing Complex Exchange Patterns in Agent Systems

OfferNets is a research initiative aimed at creating a radically decentralized economy powered by diverse, independent, interacting agents. It combines two R&D paths that are tightly related yet embrace different levels of abstraction:

- A massively scalable computing model and a software framework supporting asynchronous execution of heterogeneous processes concurrently using a shared data structure and able to model any mixture of emergent and controlled coordination among them.
- A decentralized economy providing an alternative to purely currency-based exchanges. This economy features a complex network of interactions and optimizes reciprocal exchanges of goods and services by finding agents with compatible and complementary preferences and coordinating their interactions.

Research and development of (B) is crucially dependent on (A), but the importance and application of (A) are much broader than those of (B). This means that we are designing the decentralized computing and simulation modeling platform to be maximally horizontally scalable beyond applications for OfferNets economy.

6.3.1 Decentralized Computing

The concept of open-ended decentralized computing is being developed within the OfferNets research initiative. It allows heterogeneous asynchronous processes to achieve spontaneous or guided compatibility via indirect communication through a shared topological space.

The goal of this model is to create large-scale simulations of decentralized systems, including economies, different combinations of governance regimes and structures, multiple currencies, barter networks, and more. Even though the model is inherently decentralized, it does allow for the implementation of different levels of centralization.

6.3.2 Simulation Engine

The software architecture on which OfferNets simulations are run consists of two large parts:

- An actor framework for powering asynchronous execution of heterogeneous agents' logic and peer-to-peer interactions via message passing
- A graph database powered by an enterprise-level graph database server for keeping and updating the topology information of the network and enabling indirect communication

6.3.3 Monitoring and Analysis Engine

Simulation modeling requires the collection and analysis of information about events happening in the system. Since a decentralized applications framework by definition does not have a single point of access to the system, we have built a specialized engine for collecting and handling large amounts of streaming data coming from many sources.

The basic principle of the engine is that is issues monitoring messages on behalf of each agent and then caches and indexes them into a single (but possibly distributed) database. The technical basis of the engine is ElasticStack, an integrated streaming data-management and analysis solution.

The monitoring pipeline is fully distributed, can be scaled to multiple machines, and is tolerant of failures and restarts of each component. Likewise, the simulation engine can be readily scaled to multiple machines depending on the required load for simulation or production environments. Both provide real-time monitoring across all machines via web front-ends. Real-time network and agent activity monitoring and event capturing are available via custom web front-ends accepting data streams from other parts of the infrastructure.

6.3.4 OfferNets Economy

The decentralized computing model and architecture allow us to implement, test, deploy, and observe the evolution of a virtually unconstrained number of computational processes interacting and coordinating directly or indirectly within the ecosystem. The challenge is to define concrete processes, design their interaction, and fine-tune the OfferNets economy toward preferred dynamics.

The OfferNets domain model is specified as a property graph schema, meaning it is formalized in terms of types of nodes, their properties, types of edges, their properties, and processes defining graph-traversal and mutation constraints. Every agent operating in the network is allowed to implement any process. Processes that require interaction with the social graph of OfferNets are implemented as graph traversals. Other processes are represented as conventional asynchronous algorithms. OfferNets currently implements basic similarity-search and cycle-search processes. Cycle execution and advanced search processes are in the pipeline.

Conceiving, implementing, and running computational experiments on OfferNets and then analyzing the data collected by the monitoring engine is computationally intensive due to the large parameter space and many repetitions needed for meaningful exploration.

Furthermore, it is an open-ended process in the sense that every simulation raises fresh questions and informs the setup of the next one. Both the computational infrastructure and the domain model change and improve with each iteration. Since the start of simulation modeling experiments in August 2018, we have been running simulations to compare centralized and decentralized search algorithms on the same graph structures.

6.3.5 Integration into Main SingularityNET Network

OfferNets research initiative is estimated to contribute to the development of SingularityNET's decentralized AI network in a few different ways. The exact avenues and scope of integration will be determined after the beta launch from the following possibilities:

- *Developing an automated decentralized marketplace and economy of AI agents on top of the SingularityNET network*. Since OfferNets provides a generic decentralized mechanism of chaining processes by their inputs and outputs, it may be a basis to research the mechanism of building automatic workflows among SingularityNET AI agents.

- Large-scale *simulations and testing on top of SingularityNET beta, including simulations of reputation systems*. OfferNets allows us to run simulations on top of SingularityNET's beta infrastructure by constructing simulated AI service providers with different behaviors that would call SingulartyNET infrastructure by its gRPC API.

- *Providing conceptual and computational insights into the operation and governance of decentralized networks for implementation in SingularityNET*. OfferNets follows a radically decentralized design philosophy, but real-world systems often require a healthy balance between centralization and decentralization. Pushing the limits of decentralization in a research environment provides insights that may be useful in pragmatic settings, yet may have to be separately tested and implemented.

- Further conceptual research into and simulation modeling of new decentralized economic and social models that maximize the capabilities of AI and advanced autonomous robot integration in SingularityNET and possibly beyond.

This avenue can be extended to include partnerships with existing think tanks working on new economic and social governance ideas.

7. Guiding the Network from Infancy to Childhood and Beyond

The specifics of the SingularityNET platform and the assemblage of AI algorithms and services available on the network are intended to evolve and adapt in an agile way based on the needs of the ecosystem and the contributions of the community. The success of the project will entail the rapid obsoleting of a significant percentage of the material presented in this whitepaper.

The spirit of the SingularityNET design, however, is intended to be robust with respect to growth and change. The concept of a network of interoperating and value-exchanging AIs, controlled in a democratic and decentralized manner, delivering services to customers on their own and interlocking into subnetworks whose intelligence exceeds the sum of the intelligences of the parts—this is the key and what the project founders and the guiding SingularityNET Foundation wish to see continue even as the particulars of protocols, algorithms, structures, and standards mature.

A consequence of this dynamic aspect of a project like SingularityNET is the centrality of the community to the project. It is the ecosystem of developers, users, testers, evangelists, and other community members that will drive the ongoing growth and change of the platform and the AI it supports.

With the beta release, SingularityNET is beyond its infancy but still in its very early childhood. In this germinal phase, the growth of the network will be powered and guided by the exchange of information, passion, and value among both the humans and the AIs involved in the network.