Intelligent Agent for Web Watching : Belief System and Architecture

2014, Anton Kolonin[1]
[1]Aigents Group

Abstract. In this paper we describe development of intelligent software agent for monitoring information at the web in favor of human users. We discuss construction of agent belief ontology, present its communicative interface, describe its behavioral modes and implementation architecture and follow up with a testing approach.

Keywords: agent; belief; conversation; experiential learning; monitoring; natural language; ontology; robot; semantic web; Turing test; web search.

Introduction

There are multiple complementary trends in the modern information technology development such as advances in predictive search [1], creation of artificial general intelligence agents [2], raise of market of intelligent consumer devices or the so-called “Internet of Things” [3]. All these converging technologies mean wide spreading of semi-intelligent artificial software agents embodied in various software services and utilities as well as hardware consumer electronic devices, interacting with one another and their human masters. At the same time, the interaction is assumed to be carried out in the context of knowledge structured by means of “Semantic Web” technology, where each of the agents has its own belief ontology while all communicating agents share some common foundation ontology.

In such “Internet of Things”, artificial and real human agents are talking about various “things”, representing semantic entities with meaningful relationships between them, possibly including other agents. Then, in believes of the peer agents, the agents are “**things**” themselves, so things (cars, refrigerators, thermostats, computers, smartphones, people) are “**talking about things**” - everyone about each other.

In the following research we describe result-oriented development of a particular software agent specialized in statically watching or dynamically surfing the web, monitoring specific web resources and looking up for a topic of interest given by its human master. Starting with the requirements for an agent, we construct its belief ontology, present its communicative interface, design its internal architecture and briefly discuss a testing approach.

For this work we will be assuming the minimum capability of any agent of the kind would be to maintain a bilateral conversation with a partner (for instance, human) using a simplistic semi-natural language transported as a plain text over any protocol such as TCP/IP, HTTP, IRC, email, etc. Having this provided, the same language can be used as a communication tool by itself or as a protocol for API used to create a top-level graphical user interface or enable to build a speech interface employing third-party text-to-speech and speech-to-text technologies.

Requirements

Our goal is to design and implement a robotic agent operating in the World Wide Web, performing web searches, monitoring specific topics and reporting news related to them – all on behalf of its human owner. Such an agent should be capable of knowing the list of sites or web pages to watch for the target data, as well as the list of topics to be tracked. The patterns or templates to be used to this purpose can be learned by an agent in the course of experiential learning or pre-set by a human operator. The agent would also need to have a list of its peer contacts that have to be updated with collected news.

The sensing capabilities of such an agent or a robot would get represented by an internal time sensor plus variables keeping the context specific to processing web resources (URLs of the sites and downloaded web pages) and other variables to keep the facts (concepts and relationships) extracted from the resources for these topics. There could also be a variable to supplement

reward/punishment, so that templates as well as behavioral patterns can be learned by an agent by the trial-and-fail method with a feedback from the master. The acting capabilities of this agent, besides communication with human peers, would include downloading HTML pages from the list of sites, matching templates in the pages and extracting specific values of interest from the findings. Briefly, an agent should be capable to carry out the following activities.

- Get familiar with new personalities (human **users**), represented by names (to personify), email addresses (to send notifications to), some specific information (like date of birth – in order to resolve full namesakes) plus some secret information (to confirm identity).
- Establish verbal (chat) conversations with human users (and possibly other agents of the kind) having the identity of the peer confirmed by secret information provided.
- Provide an ability to recall or reset the secret information (if forgotten) by email (if supplied).
- Accept specification of some number of the web **sites** of interest provided by a human participant of the conversation.
- Accept specification of some **things** interesting for human participant of a conversation, associated with these sites.
- For any thing of interest, optionally specify some textual **patterns** indicating occurrence of these things in the web site text.
- For the patterns, be able to manually configure indicative combinations of keywords/tokens (e.g. “house sale”), variants and lemmas (e.g. “large”, “huge”, “big”, “bigger”, “biggest”, etc.) and variable placeholders (e.g. number, date, text).
- Keep monitoring all sites of interests and respective things given by all familiar users and, if any new findings are discovered, provide users with **news** updates by chat (if there are open conversational sessions) or email or SMS/text messages, providing information about the time, site, particular thing and textual context of its experience.
- Be able to obtain feedback from a user supplied with the news in respect to relevance and novelty of the news – so the user can either confirm relevance (e.g. “good news!”) of the information or agree with relevance but deny novelty (e.g. “good, but don't show it again anymore”) and finally deny any relevance at all.
- Be able to learn from user feedback and infer the extensions of the user's pattern on its own, so the former can eventually be overridden with the inferred pattern if it provides more relevant and novel news.
- Maintain a conversation with users letting them explore the sites and things of interest being operated by an agent, and let users to amend them – so the user can ask for lists of sites, things and links between them, add, remove or amend sites and things and their properties.
- Maintain a conversation with users retaining properties specific to them such as notification frequency (hourly, daily, weekly, monthly) and the time to send the news.
- For the operations described above, keep users privacy, isolating their private data from the others’ data so all addition, removal and amendment operations are applied to the image of a human user in the agent's belief system only – for each of the human users the agent is familiar with.

To support all of the above, the following belief ontology of an agent could be constructed.

Belief Ontology

In order to construct agent's domain-specific ontology, first we define a foundation ontology used to express everything else. To describe the agent's internal belief ontology we use the same language that will be used by the agent for its interactions with humans and other agents. For this purpose we could use some “interlingua” language such as ORL [4] or Lojban++ [5] and we finally selected Agent Language (AL) specially designed for the purpose of the project (to be precise, English dialect of AL will be used). A detailed description of the language goes beyond the scope of this paper; however, it will be presented in the examples below. Briefly, the language grammar can be

seen as Turtle [6] notation extended to deal with lists of arguments in place of subject, predicate and predicate-object slots, with possibility to build complex chains of predicates and operate with disjunctive (indicated by parentheses), conjunctive (indicated by braces) and successive (indicated by brackets) lists of arguments.

First of all, we assume that any **thing** (semantic entity) must have a unique **id** (**owned** by the entity) and possibly may have one or more **names** (potentially **shared** with other namesake things). Further, we rely on such semantic relationships between things as "**is**" (being an instance of something), "**has**" (possessing certain properties) and "**does**" (be capable of doing specific actions), as expressed in AL below. The semantic relationships are represented by **properties** (effectively typed semantic links or ternary relationships [7]) which can be potentially assumed **obligatory** for a thing (so the thing must have at least one relationship of a type). Also, some of the properties may **reflect** others being reverse to them by meaning. Bold text and capitalization in the following statements do not convey syntactic meaning and are used solely for the illustration purposes, distinguishing terms in subject, predicate verb and predicate object roles.

- Thing **has** Id, Name, Is, Has, Does, Times.
  - Id **is** Property, Owned, Number, Obligatory.
    - Property **has** Type, Source, Target.
      - Type, Source, Target **is** Thing.
  - Name **is** Property, Shared, String.
  - Is, Has, Reflects **is** Property, Shared, Thing.
  - Does **is** Property, Shared, Action.
  - Action **is** Thing, Executable.
  - Times **is** Property, Shared, Time.
    - Time **is** {Today, Yesterday, Tomorrow, Date-time, Date, Month, Year}.
      - Date-time **has** Daytime, Date.
- Time **has** Events.
  - Events **is** Property, Shared, Thing, **reflects** Times.

In terms of object-oriented design, the **is/has/does** relationships identify such relationships as inheritance (and opposing instance), attributes (or member variables) and methods (or member functions) - respectively. It should be noted that, unlike many other ontologies, we do not attempt to distinguish different kinds of inheritance (such as inheritance and instance) explicitly, so that instances of classes and objects are all subclasses of one generic abstract **thing** [4]. Also, we assume the **action** is just a specific **executable** kind of **thing**, with the lifespan restricted by the execution time and runtime variables being member attributes. The meaning of other things such as **name**, **number, string, daytime** and **date** involved below should be obvious.

On the basis of the foundation ontology described above, we can construct the following domain ontology.

- Agent **is** Thing, **has** Peers (**is** Property, Agent).
  - Agent **has** Feels (**is** Property, Shared, {Good, Bad}).
- Self, User **is** Agent.
  - User **has** Surname, Birth date, Email, Secret question, Secret answer, Update time, Update period, Things, Shares, Likes.
  - User has Sensitivity threshold (is Percentage (is Number, is {0,1,...,99,100})), Seeing shares (is Toggle (is {On, Off})), Keeping days (is Number), Basic privacy (is Toggle), Check cycle (is {Hour, Day, Week, Month}), Update time (is Daytime), Telling news (is Toggle), Emailing news (is Toggle).
    - Surname, Email, Secret question, Secret answer **is** Property, Shared, String.
    - Birth date **is** Property, Shared, Date.
    - Update time **is** Property, Shared, Time.
    - Update period **is** Property, Shared, Period.
    - Email **is** Property, Shared, String, Obligatory.

        - Things **is** Property, Shared, Thing.
        - Shares **is** Property, Shared, Thing.
- Site **is** Thing, **has** Links (**is** Property, Shared, Site).
    - Thing **has** Users (**is** Property, Shared, User, **reflects** Things).
    - Thing, Site **has** Patterns, Times, Users.
        - Patterns **is** Property, Shared, Pattern, Obligatory, {String, Lemma, Frame, Thing}.
    - Pattern, Lemma, Frame **has** Patterns.
    - Site **has** Topics (**is** Thing).
    - Things **has** Origins (**is** Site).
    - Topics **reflects** Origins.
- User **things** Site.

There are some key points in the ontology worth specific mentioning. Fist of all, we will assume that an agent would be able to maintain internal belief of a peer it is interacting with (be it a human or another computer agent), so the **things** property is used to connect a **user** (representing an agent's peer) to any thing in the agent's self mind. That is, the belief subgraph of a user can be built using that property. Next, for the simplicity of further implementation, we will associate the context of communications between an agent and its peers into a belief sub-graph having it to keep all what is typically maintained in communication sessions (it means that no more than one communication session can be established between an agent and its peer). Specifically, the interaction graph for a user would be represented by a behavioral schema of its **does**-property subgraph, described in the implementation section below.

The part of the agent's ontology dedicated to operations with text can be described as **pattern**, which is a fundamental entity connecting things in the ontology with their textual representations or templates. In general, pattern can be anything starting with an individual symbol or glyph and ending with a complex hierarchy of other patterns. In our design, we consider that pattern could be either a text **string** (of 1 or more characters or glyphs), a **lemma** (either linguistic lemma representing a base word plus all its inflections or psycholinguistic lemma representing a set of synonyms conveying some particular meaning), a **frame** (a set of patterns to be stated in order, as defined per [8]) or any thing placed in the parent pattern as a variable to be resolved by pattern instance in a real text. Notably, the lemma and frame can have patterns so hierarchical textual templates are possible.

Communication Interface

When an agent possesses the belief system described above, it enables supporting the following communication scenario, expressed with the Agent Language grammar. The following dialogues denote a human user as **H:** and agent as **A:**. Italic text is used to mark omittable parts of textual communication, which can be inferred by the conversational context. On the right side, comments after double slashes indicate what is going on during the conversation.

H: *My* login.	// Login
A: *Your* name, surname, birth date, email?	// … possible identifiers asked
H: *My* name John, email john@doe.org.	// … unique ones supplied
A: *Your* secret question, secret answer?	// Registration
H: *My* secret question “pet name”, secret answer “Toby”.	// … combination retained
A: *Your* pet name?	// … “password” requested
H: *My pet name* Toby.	// … “password” matched
A: Ok.	// … “password” confirmed
…	// Conversation
H: *My* logout.	// Logout
...	
H: *My* login.	// Login
A: *Your* name, surname, birth date, email?	// … possible identifiers asked

H: *My* email john@doe.org. // … unique ones supplied
A: *Your* name John? // Verification
H: Yes. // … disambiguation made
A: *Your* pet name? // … “password” asked
H: *My pet name* Toby. // … “password” entered
A: Ok. // … identity confirmed
… // Conversation
H: *My* secret question, secret answer? // Verification change
A: *Your* verification code? // … email with code sent
H: *My verification code* 12345-67890. // … entered code from email
A: *Your* secret question, secret answer? // … re-enter identity information
H: *My* secret question “Pet name”, secret answer “Toby”. // … combination retained
A: *Your pet name* Toby? // … “password” checked
H: Yes. // … “password” matched
A: Ok. // … change confirmed
… // Conversation
H: *My* email doe@john.org. // Email change
A: *Your* verification code? // … email with code sent
H: *My verification code* 12345-67890. // … entered code from email
A: Ok. // … confirmed new email
… // Conversation
H: *My* sensitivity threshold 80, update time 9:00. // … setting user properties
A: Ok. // … update confirmed
H: *My* check cycle, keeping days? // … retrieving user properties
A: *Your* check cycle day, keeping days 10. // … data retrieved
… // Conversation
H: *My things* site http://www.weather.com, http://www.accuweather.com .
A: Ok. // … sites retained
H: *My things* site no http://www.accuweather.com, http://weather.yahoo.com .
A: Ok. // … sites updated
H: *My things* site? // … asking for site instances
A: *Your things site* http://www.weather.com, http://weather.yahoo.com/.
H: *My things* temperature, storm. // … adding new things to belief
A: *Your things* temperature, storm has, patterns? // … asking for unresolved details
H: Temperature has value (is number), scale (is {'°C ', '°F'}), patterns [temperature scale Novosibirsk Russia]. // … properties of thing listed
A: Ok. Storm has, patterns? // … keep resolving details
H: Storm has region. Region is {Northwest, Southwest, Midwest, North Ural, South Ural, Northwest Siberia, Southwest Siberia, Northeast Siberia, Southeast Siberia, North Far East, South Far East}. // … properties of thing listed
A: Ok. Ok. Storm patterns? // … keep resolving details
H: Storm patterns [{'storm', 'storms'}, {approach, move} region]. Approach is {'approach', 'approaching', 'approaches'}]. Move is {'move', 'moving', 'moves'}.
A: Ok. Ok. Ok. // … resolving is complete
H: *My* things? // … asking count/cardinality
A: *Your things* 3. // … count/cardinality reported
H: *My* things name? // … asking name property of things
A: *Your things* name site, name temperature, name storm. // … things with name property listed
H: Temperature patterns? // … asking for patterns of a thing
A: *Temperature patterns* [{'storm', 'storms'}, {approach, move} region].
… // Conversation
H: *My things site* http://www.weather.com, http://weather.yahoo.com topics temperature.
A: Ok. // … things of topic for site are set

H: *My things* storm origins http://www.weather.com . // … setting origin site for thing
A: Ok. // … origin site for thing is set
H: *My things site* http://weather.yahoo.com topics? // … asking for things of topic for site
A: *Your things site http://weather.yahoo.com topics* temperature, storm.
H: *My things* storm origins? // … asking site origins for topic thing
A: *Your things storm origins* http://www.weather.com . // … origin site for topic thing reported
… // … pushing news
A: Storm times today, region Southwest Siberia, origins http://www.weather.com?
H: Yes. // … positive feedback, true belief
A: *Your* feels? // … indirect feedback polled
H: *My feels* good. // … implicit feedback provided
… // Conversation
H: Storm times yesterday, region Southeast Siberia. // … authoring news, true belief
A: Ok. // … new thing instances acquired
H: Storm times today, region Southwest Siberia, origins http://www.weather.com!
A: Ok. // … learning pattern
H: Storm times yesterday, region? // … polling properties of news
A: *Storm times yesterday, region* Southeast Siberia. // … property of news reported
H: Ok. // … positive feedback, true belief
H: Storm times today, region, sources? // … polling news
A: Storm times today, region Northeast Siberia, origins http://www.weather.com.
H: No. // … negative feedback, false belief
H: *You* feels good? // … checking agent's state
A: Yes. // … agent's state reported
H: *You* feels bad! // … indirect feedback pushed
A: Ok. // … indirect feedback accepted
H: No storm times today, region Northeast Siberia. // … false belief, decrement evidence
A: Ok. // … direct feedback accepted
H: *My things site* http://www.weather.com time today, yesterday topics?
A: *Your things time http://www.weather.com time today, yesterday topics* storm times today, region Northeast Siberia, Southeast Siberia. // … news topics for site provided
H: *My things* site users? // … ask for users of thing class
A: *Your things site users* Ian, Ann, Bob. // … other users of thing listed
H: Storm times today, region Northeast Siberia users? // … ask for users of thing instance
A: *Storm times today, region Northeast Siberia user*s Ian, Ann, Bob.
H: Site http://www.weather.com users name, email? // … ask for users of site (if allowed)
A: *Site http://www.weather.com users* name Ian, email ian@doe.com, name Ann, email ann@doe.org, name Bob, email bobdoe@hotmail.com. // … properties of site users listed
… // … looking up topics in peer's contexts
H: *My things site* http://www.weather.com topics (users {Joe,Pam})?
A: *Your things site http://www.weather.com topics (users {Joe,Pam})* tornado, storm.
H: *My things site* http://www.weather.com topics (time {today, yesterday}, users {Joe,Pam})?
A: *Your things site http://www.weather.com topics (time {today, yesterday}, users {Joe,Pam})* storm (times today, region Hawaii), tornado (times tomorrow, region Ohio).
… // … getting news in peer's contexts
H: Storm origins (users Joe or Pam)? // … asking to count thing site origins
A: *Storm origins (users Joe or Pam)* 3. // … number of origins/topics reported
H: *Storm origins (users Joe or Pam)* name? // … list thing site origins
A: *Storm origins (users Joe or Pam) name* http://www.weather.com, name http://weather.yahoo.com, name http://www.accuweather.com.
H: Storm (time today or tomorrow and users Joe or Pam)? // … asking to count thing news
A: *Storm (time today or tomorrow and users Joe or Pam)* 2.
H: Storm (time today or tomorrow and users Joe or Pam) times and region and origins?

A: *Storm (time today or tomorrow and users Joe or Pam)* (times today, region Hawaii, sources http://www.weather.com), (times today, region Alaska, sources http://weather.yahoo.com).

The presented level of human – agent communication, as anticipated, will not only enable a human user to control an agent monitoring the web for the user's benefit, but will also let the agent possibly learn new text patterns and semantic associations and evolve a behavioral schemata made of elementary actions instead of having them explicitly specified by the user.

Architecture

It is expected that an agent can operate as a Software-as-a-Service (SaaS) server solution or a desktop or mobile application. For any implementation, it is anticipated to have two major layers – a user interface and an engine with storage. In case of a mobile application, there would be a "lite" version of storage and a single-user engine (e.g. a single "cell" agent). In case of a desktop or a web server version, there would be storage of respective power and engine capable to host multiple users (e.g. multi-agent "farm of cells"), so that any desktop installation could operate as a server for peers connecting to it. For standalone mobile and desktop versions, communication between user interface and the engine would be carried out via an internal API in-process. For a client mobile version and web browser clients, client-server communication would be done over HTTP/HTTPS.

This way or another, we anticipate the same Agent Language can be used as a high-level protocol for all of these cases. Moreover, it can be used as a non-graphical chat interface for communication over any protocol such as TCP/IP, HTTP, ICR or via email or SMS messages. Further in this work we will focus on the internal architecture of the **agent engine** capable to communicate via Agent Language over any of the discussed transport protocols. Respectively, we will assume that a user session can be identified as TCP/IP socket, HTTP cookie, IRC nickname, SMS phone number or email address, so that the conversational context will be associated with a corresponding user.

The agent functionality will be implemented as a number of behavioral modes, each with its current state, attached either to each of the users or to the agent's self. Accordingly to the definition [3], each of the modes is a specific process, expressed as a behavioral subgraph implementing a particular activity. The process, being a persistent "executable" object, also has a number of internal variables representing its contextual state. The design is restricted by a constraint that each user or self is uni-modal, so there is only one behavioral context experienced by each of them at a time. However, it is expected that there will be lateral interactions between the modes so the self can trigger particular modes for a certain user while a user can spawn the activity of a specific mode in the agent's self. Below we describe different modes allocated to the agent's self and each of its users - with transitions possible between these modes, using the AL syntax (with **next** relationship indicating transition between modes, braces indicating disjunction transitions and parentheses – the conjunctive ones).

- User **does**
  - Interrogation, Confirmation, Declaration, Direction,
  - Login, Registration, Verification, Logout,
  - Email Change, Verification Change,
  - Clarification,
  - Feedback,
  - Conversation.
    - Login next {Verification, Registration, Login}.
    - {Verification, Registration} next Conversation.
    - Conversation next {Interrogation, Confirmation, Declaration, Direction, Logout, Clarification, Feedback, Email Change, Verification Change}
    - {Interrogation, Confirmation, Declaration, Direction, Logout, Clarification, Feedback, Email Change, Verification Change} next Conversation.

        - Feedback next (Learning Patterns, Conversation).
- Self **does**
    - Checking Times, Reading Sites, Matching Topics, Watching News, Messaging News, Learning Patterns, Forgetting Stuff.
        - Checking Times next {Reading Sites, Checking Time}.
        - Reading Sites next Matching Topics next Watching News next Messaging News next Checking Times.

Below we give a more detailed description of each of the modes.

Interrogation, Confirmation, Declaration, Direction – these user modes correspond to handling one of the four Agent Languages statements respectively.

Login, Registration, Verification, Logout – user modes implementing authentication functionality, so that authenticated context of further conversation can be established by completing either Login or Registration procedure (where Verification is used to confirm Registration) and then the context can be cleaned up by Logout at the end. Authentication workflow can potentially be executed on both sides of each communication peers establishing **mutual authentication** so that not only the server is certain that it serves the correct client, but the client is also certain that it is connected to the correct server. The purpose is to isolate the properties belonging to a user in an agent's belief and also assure the communication is established to the correct peer agent.

Email Change, Verification Change – these user modes are specific forms of Declaration augmented with extra confirmations via email (or possibly SMS) with special confirmation codes.

Clarification – it is a special user mode associating generic declarations, interrogations, declarations and directions if the statement issued by the peer communication party is ambiguous and there is a need to pose contra-interrogations (**resolving unresolved variables**) to make a statement clear.

Feedback – is a user mode evaluating user's feedback and translating it to self Learning Patterns mode.

Conversation – is a centric user mode recognizing conversational text patterns from the perspective of a current user context, dispatching the control flow to the appropriate mode and then getting the control back with an updated context. In a simple form is it a matter of recognizing respective AL statements but in reality it is also a matter of using special **patterns** indicative to other modes. The patterns are expected to be same textual patterns as discussed earlier as part of agent belief associated with things and sites, and they also can be expressed in AL syntax as an expression or set (disjunctive, conjunctive or successive) of expressions. It is also possible to apply **fuzzy pattern matching** in the cases when no applicable straight matches are found to handle the conversational context.

Checking Times – is a basic self mode periodically checking time for performing updates of sites as specified by users, and initiating a news update cycle starting with Reading Sites mode.

Reading Sites – is a self mode reading structured texts (pages, blocks, columns, paragraphs, utterances) from sites specified as news origins by a user, and passing the read data further to Matching Topics mode.

Matching Topics – is a self mode matching users's patterns from texts being read from sites, and passing the found topics of user's interest to Watching News mode.

Watching News – is a self mode tracking historical backlog of the topics matched on sites and catching any **novel** and **salient** information appeared, passing the news to Messaging News mode.

Messaging News – is a self mode dispatching news to users watching corresponding topics (indicating time, site, actual thing indicated by topic and respective text pattern framing it) and then returning the execution control back to Checking Times mode.

Learning Patterns – is a self mode triggered by user's Feedback mode so topics given positive or negative feedback can be used to **learn** or **dismiss** patterns associated with them on the origin sites.

Forgetting Stuff – this self mode is triggered periodically by Checking Times or

occasionally by Learning Patterns in order to compress the belief data operated by an agent. Primarily, this would involve keeping restricted subset of data representing agent's "attention focus" in the local cache (i.e. "short term memory" or **STM**), given the amount of available operating memory and pushing currently unnecessary data to persistent storage (i.e. "long term memory" or **LTM**). Further, this would deal with getting rid of irrelevant data or data of low importance and confidence ("garbage collection") moving it out of persistent LTM.

The agent architecture will be represented as a set of processes reflecting communications with the external world and agent's modes plus dedicated components for handling AL messages, web sites, session contexts and data persistence.

There will be processes bound to the agent's self, extending the basic **Selfer** process, implementing respective self modes. Further, there will be processes to run in the context of Selfer, implementing user modes, extending the basic **Conversationer** process. Finally, there will be processes in the context of Selfer, to deal with communications, extending the basic **Communicator** process, such as **Cmdliner**, **Emailer**, **SMSer**, **IRCer**, **TCPer** and **HTTPer** – implementing respective communication channels and protocols.

In order to parse AL inputs and generate AL outputs there will be respective **Reader** and **Writer** components translating data flow from Communicator to Conversationer and back. To perform proper contextual disambiguation these components will be referring to Storager component providing the content to get the actual belief ontology data and to Sessioner component to adjust communication to the context of a specific agent's user.

There will be **Siter** component holding a cache of the page data indexed by site URLs and time stamps. It will be serving Selfer to implement in-depth and in-breadth reading of web site pages and keeping track of changes on these pages over the time. The Siter will be using the **Storager** component for persistent storage of the site-specific information.

The **Sessioner** component will be keeping session contexts for all agent users across different communication channels, serving the processes of Communicator and Conversationer as well as Reader and Writer components. It will be also using the Storager component for persistent storage of session contexts.

Finally, the **Storager** component will be providing the STM and LTM data storage for all processes and components described above. It is expected the LTM storage will be implemented as a graph database or a graph-enabled interface on top of a relational or non-relational database, while the STM will be effectively a cache of persistent LTM data. The important requirements for the Storager would be ability to deal with **hyper-graphs** (so not only nodes but also links as well as entire sets of nodes and/or links can be linked in a graph) and to control the cached data scope (assuring consistency of the "attention focus").

Testing Approach

Looking ahead to a testing approach, there is an intent to apply **baby Turing test** [9], simplified to the AL grammar restrictions. The "baby Turing test" is known to be more complex than the basic "Turing test" in a sense that a computer system undergoing the test should not just be able to perform like a human. Indeed, it has to learn the level of behavioral complexity, in terms of any given language, experientially from the "ground zero". Our goal is essentially the same – given the limited terms of foundation ontology such as "thing" (used as a marker pointing to anything) and "site" (used to represent the outer would), enable the system to discuss any topics pointed at in the World Wide Web environment. The simplification will be made so that the end goal will be not to learn the entire human world belief along with full-scale natural language acquisition. Rather, we restrict the level of linguistic capabilities to achieve the level of a 2-3 year baby or a foreigner who has learned an alien language in a narrow practical domain for 1-2 months – with a substantial vocabulary but a limited knowledge of grammar reduced to very simple linguistic forms.

While practical implementation of the described agent can be done in many forms and come up in variety of graphical user interfaces connected to an engine by means of AL protocol, for the

proof-of-concept purpose we are going to implement a simplistic chat-based interface. With that kind of interface, it will be equally possible to construct agent's belief manually in the course of a human-agent dialogue, load it with some pre-configured ontological and factual data and perform experiential training through positive and negative feedback. After all, more complex interfaces and distributed multi-agent configurations could be built on top of the same interface.

Conclusion

Given the proposed foundation belief ontology, it is seemingly possible to extend it to any complex beliefs in various practical domains, enabling user to specify targets to watch on the web as well as specify explicit matching templates and provide feedback to an agent – so the latter can use artificial general intelligence techniques to evolve the desired behavioral schemata in the course of experiential learning. A simplified semi-natural "agent language" based on that ontology and suggested for communication between a computer software agent and a human user seems compact yet expressive enough for transmission and visual comprehension, easy to read and write for an average human (without special computer knowledge) and easy to parse into semantic graph operations for computer programs. The design described above is expected to be practically verified in further work, with real implementation and testing of the designated agent.

References

[1] B. Kobayashi: Law's Information Revolution as Procedural Reform: Predictive Search as a Solution to the 'In Terrorem' Effect of Externalized Discovery Costs (October 9, 2013). George Mason Law & Economics Research Paper No. 13-56. Available online at SSRN: http://ssrn.com/abstract=2340068 or http://dx.doi.org/10.2139/ssrn.2340068

[2] M. Looks, B. Goertzel, C. Pennachin: Novamente: An integrative architecture for general intelligence. AAAI fall symposium, achieving human-level intelligence, 2004, Available online: http://www.aaai.org/Papers/Symposia/Fall/2004/FS-04-01/FS04-01-010.pdf

[3] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami: Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems 29, Elsevier, 2013, Available online: http://www.buyya.com/papers/Internet-of-Things-Vision-Future2013.pdf

[4] A. Kolonin: Object-Relational Language and modern tradeoffs in software technology. International Andrei Ershov Memorial Conference, PSI 2006, Program understanding workshop, Russia, Novosibirsk, June 2006, Available online: http://www.webstructor.net/papers/2006ObjectRelationalLanguageAndModernTradeoffsInSoftwareTechnology.pdf

[5] B. Goertzel: Lojban++: An Interlingua for Communication between Humans and AGIs. Proceedings of 6th International Conference, AGI 2013, Beijing, China, August 2013, Available online: http://goertzel.org/agi-13/Lojban_paper_v1.pdf

[6] World Wide Web Consortium: Turtle – Terse RDF Triple Language". November 2012. Available online: http://www.w3.org/TR/2012/WD-turtle-20120710/

[7] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks: The Semantic Web: The roles of XML and RDF. IEEE Internet Computing. 15 (3), pp. 63-74. October 2000, Available online: http://www.cs.vu.nl/~frankh/postscript/IEEE-IC00.pdf

[8] A. Kolonin: High-performance automatic categorization and attribution of inventory catalogs. Proceedings of All-Russia conference "Knowledge – Ontology – Theories" (KONT-2013), Novosibirsk, Russia, October 2013, Available online: http://webstructor.net/papers/Kolonin-HP-ACA-IC-text.pdf

[9] B. Partee et al.: Report of Workshop on Information and Representation. Washington, DC, March 30-April 1, 1985, Available online: http://files.eric.ed.gov/fulltext/ED261533.pdf