CENode: Enabling Human-Machine Conversations at the Network Edge

William Webberley*, Alun Preece*, Dave Braines†
*School of Computer Science and Informatics, Cardiff University, Cardiff, UK; contact email: PreeceAD@cardiff.ac.uk
†Emerging Technology Services, IBM United Kingdom Ltd, Hursley Park, Winchester, UK

***Abstract*—As human-machine communication systems, such as Siri and Google Now, become more ubiquitous, the more frequently these technologies are observed and become necessary at the network edge or away from a centralised knowledge base (KB). Such systems might be particularly useful in disaster and coalition response situations. The availability of agents equipped with the ability to understand human input, maintain a conceptual model of the 'world', and report relevant knowledge in a machine and human-readable format when necessary is essential in these situations so that functionality can continue to be provided even without a network connection. We present CENode; a system comprising a KB capable of modelling entities and their relationships, and an agent that enables humans and other agents to update and query this KB directly using ITA CE (Controlled English). CENode instances are capable of being deployed in a variety of settings, such as within web applications and as standalone apps running on handheld devices or servers. We show how policies, written in CE, govern aspects of CENode behaviour in determining under what circumstances a node instance communicates with the agent of another node. The paper highlights examples of its use when deployed in coalition scenarios.**

I. Introduction

The CE Store [1] provides a processing environment for ITA Controlled English (CE), including domain modelling, inference, and natural language (NL) to CE interpretation. The CEStore, implemented in Java, is based on a client-server architecture where CE processing is performed on the server side, and clients access the capabilities of the environment via a rich set of application programming interfaces (APIs).

In this paper we introduce CENode[1], a lightweight CE processing environment implemented in JavaScript so as to be easily deployable in a variety of contexts, including web browsers, mobile apps, and servers[2]. CENode is lightweight in the sense that it does not aim to be a fully-fledged CE engine — for example offering only limited inference and NL processing — and requires relatively little network bandwidth to download and operate. Once loaded, a CENode instance can function independently without any network connection, maintaining a local KB and communicating with other CENode instances only when connectivity is available, via the CE Card conversational protocol [2] and blackboard mechanism. This makes it well-suited to deployments at the network edge, and in coalition settings where a centralised client-server model is not the most appropriate configuration.

[1]http://cenode.io
[2]For example, via Node.js: https://nodejs.org

II. Key Features and Benefits

CENode instances can either be run independently or as part of a multi-node system. In a multi-node system, at least one of the nodes needs to be run as a service (e.g., via Node.js). All CENode instances in a multi-node system are, by default, equal in terms of functionality and behaviour. This is the case even if each node is deployed in a different way (e.g., some nodes may be running as a service, some as a web application, and some as a programmatic JavaScript application). Providing information to (and retrieving information from) a node is always done via CE. Using CE as the only means of communication enables support for distributed systems including humans, CENode agents, and a CE Store.

CENode is intended to offer a number of key benefits in a coalition edge-of-network setting:

- Users have access to, and can interact with, a CENode agent directly on their device. Any CE provided to the agent can be parsed locally and any local knowledge stored can later be relayed ('told') to other agents once a network connection is (re)established.
- Because the local node is a CE processing environment, features such as CE 'autocorrect' and 'spellchecking' can be provided at no bandwidth cost and in the absence of a network connection. The local agent can quickly check the validity of any CE as it is being typed in order to guide the user towards inputting correct CE and also giving insight into the concepts and instances stored in the local CE model.
- Local NL processing of input means that only validated CE is transmitted between nodes, at a saving of bandwidth and time.
- Instead of relying on a single CE Store server with a centralised knowledge base, CENode supports a network of peers with different local knowledge base variants. This is particularly important in a coalition context where different partners may hold different knowledge.

The controlled natural language grammar understood by CENode has been extended from standard ITA CE, supporting various 'shorthands' for easier input and querying of information to and from the KB. Input made in this way (as with the NL processing) can be guided by the node's own KB, and predictions for intended sentences can be provided. Whilst not standard CE, CENode's understanding of the grammar means that the following types of sentences

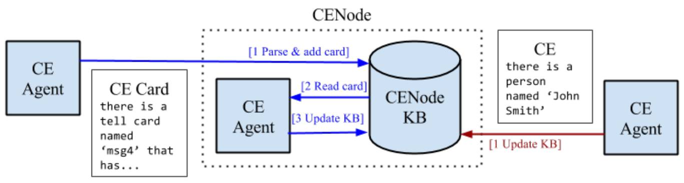

Fig. 1. Manipulating a node's KB - *left: through cards; right: directly.*

can safely be sent within CE cards to a CENode agent. For example, the CE sentence
`the teacher 'Mrs Smith' teaches the subject Computing.`
can be re-written more concisely as:
`Mrs Smith teaches the subject Computing.`

Another useful 'shorthand' is the ability to ask questions to provide users and agents with the ability to make who/what/where queries of the node's KB. As well as supporting questions such as 'what is?' and 'who is?', the interface can be used to query about relationships and properties. For example, the query
`what is teaches?`
results in the response:
`'teaches' describes the relationship between a teacher and a subject.`

III. Manipulating the Knowledge Base

Each CENode comprises a KB and a local *CE agent* that maintains the KB, shown in Figure 1. A CENode will try to process and update its KB when any CE is received. As with the CE Store, CENode instances also support the blackboard architecture, which enables users and agents to submit CE sentences wrapped in CE Cards that are addressed to the local agent. If a card addressed to the agent is received, then the agent can find the card and read it. If the card contains valid CE, then the agent can then use this to modify its KB. If the card is not addressed to the local agent, then it will remain in the KB unread. The addressee node may eventually find this card as a result of policies (see Section IV).

For example (illustrated by Figure 1), assuming a node's local agent is named `agent1`, the following two sentences received by the node would have equal effect:
`there is a tell card named 'msg1' that is to the agent 'agent1' and has 'there is a subject named Computing' as content.`
`there is a subject named Computing.`

The CE agents identified in Figure 1 represent any entity that is able to emit CE and communicate with the node. These might be human agents inputting information through a text messaging interface, or machine agents which are communicating with the node as a result of policies (see Section IV). CENode provides RESTful and programmatic APIs for supplying CE. The APIs are exposed to JavaScript applications (e.g., within web apps or Node.js applications) and the RESTful endpoints are exposed when CENode is run as a web service (e.g., again via Node.js).

IV. Agents and Policies

Each CENode instance includes a local agent (see Figure 1), which is normally responsible for updating the local KB when cards are received. Agents in multi-agent setups are also able to send cards with respect to *policies*. Policies are instructions, written in CE that, when applied to a node, may cause the local agent to try and communicate with another agent.

For example, consider a *tell policy*, which instructs the agent to forward any *tell card*s received on to another target agent and is useful for propagating information through a network of node instances:
`there is a tell policy named 'p1' that targets the agent 'agent2'.`
Other policy types include a *listen policy* (for retrieving cards from other agents) and a *feedback policy* (for governance over responses provided to received cards).

If policies are active on an agent, but there is no network route to other nodes, then the local node will still function as normal in the meantime, but will attempt to re-establish connections with other nodes once a network becomes available. Combining policies in different ways allows for the deployment of various network topologies of nodes that might be useful in different coalition settings.

V. Conclusion

In this paper, we have briefly introduced CENode and described its key benefits in distributed and edge-of-network scenarios. Initial experiments with CENode have demonstrated much promise for supporting rich human-machine interactions entirely locally, and have also illustrated the power of deploying multiple CENode instances in a network for crowdsourced observation exercises. Given the initial success, we plan to develop CENode further and use it as a basis for additional experiments and scenarios that require human-machine conversational interactions for solving tactical and crowdsourced intelligence, surveillance, and reconnaissance tasks.

Acknowledgement

This research was sponsored by the US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defence or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] D. Braines, D. Mott, S. Laws, G. de Mel, and T. Pham, "Controlled english to facilitate human/machine analytical processing," in *Proc Next-Generation Analyst (SPIE Vol 8758)*. SPIE, 2013.
[2] A. Preece, D. Braines, D. Pizzocaro, and C. Parizas, "Human-machine conversations to support multi-agency missions," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 18(1), pp. 75–84, 2014.