Coalition C3 Information Exchange Using Binary Symbolic Vectors

Chris Simpkin,Ian Taylor,
Alun Preece
Cardiff University
simpkin.chris@gmail.com
taylorij1@cardiff.cf.ac.uk
PreeceAD@cardiff.ac.uk

Graham A Bent, Richard Tomsett
Raghu K Ganti
IBM Research
{gbent,rtomsett}@uk.ibm.com
rganti@us.ibm.com

Olwen Worthington
Dstl
{olworthington@mail.dstl.gov.uk

***Abstract*—Communicating information between entities requires there to be a common understanding of the language and the concepts being exchanged. Where information has to be rapidly conveyed, or where the communications are limited, the use of specialized language or symbols is often used but this requires the various parties to know the precise meaning of the terminology (i.e. the parties can correctly interpret the language or symbols used). In this paper we propose a new potential approach for information exchange that makes use of binary symbolic vectors to semantically represent the information to be exchanged. We show how such vector representations can be used to semantically convey large amounts information between coalition partners in a compact representation that can be interpreted at different levels of semantic abstraction. In this paper we illustrate the concept by showing how a complex text (in this case Shakespeare's play Hamlet) can be represented using symbolic vectors. We discuss how such representations can be efficiently communicated between coalition partners and how this can significantly reduce the communications requirements. We develop the concept to illustrate how symbolic vectors could be used to perform command and control functions in distributed decentralized coalition operations.**

I. Introduction

Communicating information between entities requires there to be a common understanding of the language and the concepts being exchanged. Language itself is shared meaning using an agreed set of symbols to interact and communicate with others and applies equally to human-to-human and human-to-machine communications. Where information has to be rapidly conveyed, or where the communications are limited, the use of specialized language or symbols (e.g. jargon, NATO Joint Military Symbology) speeds up the process but requires the various parties to know the precise meaning of the terminology (i.e. the parties can correctly interpret the language and/or symbols used). Whilst the use of such specialized terminology works well within a single organization or group, when communication occurs between a diverse set of organizations, such as in coalition operations, this often leads to misunderstanding and confusion and especially when the same symbols can have significantly different meaning. There needs to be some way of ensuring that there is common understanding, which introduces the broader notion of distributed cognition.

In [12] we proposed the idea of a *distributed federated brain* envisioned as a distributed Cognitive Computing System (CCS) that consist of humans and software that work together, with the computer systems showing characteristics of a human brain to assist humans in an intuitive manner. Such a system requires a means for communicating information between the entities in a way that can be unambiguously processed.

To illustrate how a distributed federated brain concept might be realized, we have been investigating how future analytic applications can be constructed from component services, in a dynamic, decentralized manner. The approach is particularly applicable to a future military Internet of Battlefield Things (IoBT) may comprise hundreds or even thousands of devices and component services that need to be dynamically linking the services into different workflow configurations to create the applications necessary support mission needs. In a coalition setting the sensors and services will have been developed and owned by different coalition partners and standard centralized approaches, using formal ontology's to facilitate service definition and service matching for configuring applications are unlikely to work. In [10, 11] we have shown that service definition, service matching and decentralized service composition can be achieved using an approach based on Vector Symbolic Architectures (VSAs) and that the approach offers significant advantages in environments where communication is unreliable.

In the context of this paper, we argue that in addition to supporting decentralized service composition, that symbolic vector representations can be used to address a number of the key inter-operability issues. The first issue relates to the communication of information between coalition partners in a way that minimizes communications bandwidth. We show that using symbolic vectors enables a highly compact semantic representation of the information can be generated and that this can be interpreted by the receiving entities, even if they do not have an exactly matching representation. The second issue relates to the complex challenge of how to perform coordinated missions that require multiple inter-related sub-tasks to be performed by coalition partners, where the partners can perform the required sub-tasks but might do so in a slightly different way to the way own forces would perform the same

task.

To demonstrate the communication concept, we show how a complete vector representation of Shakespeares play Hamlet can be constructed as a hierarchy of binary symbolic vectors. In this representation, at the highest semantic level a single vector semantically represents the whole play. This vector is unambiguous in that it is a superposition of vectors that capture the semantic meaning of the actions performed at lower levels of the play down to the individual words. We show how the semantic meaning is preserved at different levels of the hierarchy even when comparing different editions of the play.

To demonstrate the command and control concept we again use the Hamlet representation. We argue that running a coalition military operation, where different coalition units and resources are used, is analogous to a distributed service workflow. Using the Hamlet play as an analogue of a military mission, we explain how the individual roles performed by the actors are the equivalent of different coalition partners performing the sub-tasks required to complete the mission. We show how by simply exchanging semantic vectors a mission can be performed.

In section II we describe some of the related work in this area; In section III we describe some of the properties of VSAs, how they are used to create semantic representation and explain how higher levels of semantic representation can be represented using symbolic vectors; In section IV we describe how a symbolic vector representation of Shakespeares Hamlet can constructed and how the resulting vectors can be used to communicate the play (Section IV-A) and as a mechanism for command and control (Section IV-B); In section V we present our conclusions and direction for future work.

II. Related Work

VSAs are a family of bio-inspired methods for representing and manipulating concepts and their meanings in a high-dimensional vector space [5]. They are a form of brain like distributed representation that enables large volumes of data to be compressed into a fixed size feature vector such that the semantic meaning of the data and relationships that they represent is preserved. Such vector representations were originally proposed by Hinton [3] who identified that they have recursive binding properties that allow for higher level semantic vector representations to be formulated from, and in the same format as, their lower level semantic vector components. Eliasmith, in his book How to Build a Brain [1], shows how these vector representations can be used to perform brain like neuromorphic cognitive processing. He coined the phrase semantic pointer for such a vector since it acts as both a semantic description of the concept, which can be manipulated directly and a pointer to the concept. As such they are said to be semantically self-describing. VSAs are also capable of supporting a large range of cognitive tasks such as: Semantic composition and matching; Representing meaning and order; Analogical mapping; and Logical reasoning [4]. Consequentially they have been used in natural language processingand cognitive modelling. One of the important properties of the recursive vector binding is that higher level semantic vectors can be unbound to recover the constituent vectors from which they composed.

In [11] we have shown how service workflows, comprising of multiple distributed micro-services, can be represented as symbolic vectors and how these vectors can be exchanged to construct the required workflow with no centralized control. We refer to this type of application construction as Instinctive Analytics in which the user specifies the workflow required and the appropriate services connect themselves together to perform the required task. The advantage of using symbolic vectors to perform these operations is that the vectors semantically represent the required services and therefore alternative service compositions can be generated. Whilst the work to date has focused on service composition for distributed analytics the concept of distributed workflow is not limited to this relatively narrow domain of service composition and has much wider implications. In the context of military operations, for example, we argue that both information sharing and command and control can be represented as types of workflow.

III. Vector Symbolic Architecture

VSAs use hyper-dimensional vector spaces in which the vectors can be real-valued, such as in Plate's Holographic Reduced Representations (HRR) [7], typically having N dimensions ($512 \leq N < 2048$), or they can be large binary vectors, such as Pentti Kanerva's Binary Spatter Codes (BSC) [5], typically having $N \geq 10,000$. For the work here, we have chosen to use Kanerva's BSC but we note that most of the equations and operations discussed should also be compatible with HRRs [8].

Typically, when using BSC, a basic set of symbols (e.g., an alphabet) are each assigned a fixed, randomly generated hyper-dimensional binary vector. Due to the high dimensionality of the vectors the basic symbol vectors are uncorrelated to each other with a very high probability. Hence, they are said to be *atomic* vector symbols [5]. Vector *superposition* is then used to build new vectors that represent higher level concepts (e.g., words) and these vectors in turn can be used to recursively build still higher level concepts (e.g., sentences, paragraphs, chapters...). These higher level concept vectors can be compared for similarity using a suitable distance measure such as normalized Hamming Distance (HD).

HD is defined as the number of bit positions in which two vectors differ, divided by the dimension N of the vector. When setting bits randomly, the probability of any particular bit being set to a 1 or 0 is 0.5; hence, when generating very large random vectors, the result will be, approximately, an equal, 50/50, split of 1s and 0s distributed in a random pattern across the vector. Thus, when comparing any two such randomly generated vectors, the expected HD will be $HD \approx 0.5$. Indeed, for 10kbit binary vectors, the probability of two randomly generated vectors having a HD closer than 0.47 (i.e., differing in only 4700 bit positions instead of approximately 5000) is less than 1 in 10^9 [5, page 143]. For the same reason; *atomic* vectors can be generated as needed, on the fly, without fear

that the newly generated random vector will be mathematically similar to any existing vector in the vector space. Further, by implication, when using HD on BSC to test for similarity, a threshold of 0.47 or lower implies a match has been detected with a probability of error $\leq 10^{-9}$. A threshold of 0.476 or lower implies a match with a probability of error of $\leq 10^{-6}$. Thus, in our experiments, we used 0.47 as the threshold.

VSAs employ two operations—i.e., *binding* and *superposition*. *Binding* is used to build *role-filler* pairs which allow sub-feature vectors to remain separate and identifiable (although hidden) when bundled into a compound vector via *superposition*. For BSCs, *binding* is a lossless operation, while *superposition* is lossy. Kleyko [6, Paper B, page 80] supplies a mathematical analysis of the capacity of a single compound vector such that it can be reliably unbound, i.e., its sub-feature vectors can be reliably detected within the compound vector. This analysis shows that for 10kbit binary vectors the upper limit of superposition is 89 sub-vectors. To encode large workflows with more complex service descriptions we require a method for combining more vectors into a single vector whilst maintaining the semantic matching properties. This is known as 'chunking'.

Chunking is a recursive binding method that combines groups of vectors into a single compound vector. The resultant vectors are then used as the basis for further chunking operations, thus, recursively producing a hierarchical tree structure as shown in Figure 1. Chunking proceeds from the bottom up so that each node in the tree is a compound vector encapsulating the child nodes from the level below. Various methods of recursive *chunking* have been described [7, 8, 5, 6]. However, such methods suffer from limitations when employed for multilevel recursion: some lose their semantic matching ability even if only a single term differs, others cannot maintain separation of sub-features for higher level compound vectors when lower level chunks contain the same vectors [5, page 148] [7, pages 61, 72, 74–para2] [6, Encoding Sequences, page 14]. In order to overcome these issues we use a hierarchical binding scheme described in [11], that uses two methods for permuting binary vectors. The resulting recursive encoding scheme generates vectors at different levels of the hierarchy that capture the semantic information from the levels below.

Recchia and Kanerva point out that for large random vectors, any mapping that permutes the elements can be used as a binding operator, including cyclic-shift [9]. This is because if a high-dimensional vector has no pattern to the distribution of its element values then one or more cyclic rotations will produce a completely different vector as compare to the original value and hence cyclic shift has the same effect of applying a random permutation, i.e., since there was no pattern in the element distribution to start with there will remain no pattern to the element distribution after rotation.

The encoding scheme shown in eq. (1) employs both XOR and cyclic-shift binding to enable recursive bindings capable of encoding many thousands of sub-feature vectors even when there are repetitions and similarities between sub-features:

$$Z_x = \sum_{i=1}^{cx} Z_i^i \cdot \prod_{j=0}^{i-1} p_j^0 + StopVec \cdot \prod_{j=0}^{i} p_j^0 \tag{1}$$

Omitting $StopVec$ for readability, this expands to,

$$Z_x = p_0^0 \cdot Z_1^1 + p_0^0 \cdot p_1^0 \cdot Z_2^2 + p_0^0 \cdot p_1^0 \cdot p_2^0 \cdot Z_3^3 + \ldots \tag{2}$$

Where

- $\cdot$ is defined as the XOR operator;
- $+$ is defined as the Bitwise_Majority_Vote/Add operator;
- The exponentiation operator is redefined to mean cyclic-shift—i.e., positive exponents mean C_{shift_right}, negative exponents mean C_{shift_left}. Note that cyclic shift is key to the recursive binding scheme since it distributes over $+$ (i.e., bitwise majority addition) and $\cdot$ (i.e., XOR) hence it automatically promotes its contents into a new part of the hyper-dimensional space; thus, keeping levels in the chunk hierarchy separate;
- Z_x is the next highest semantic *chunk* item containing a *superposition* of x sub-feature vectors. Z_x chunks can be combined using eq. (1) into higher level chunks. For example, Z_x might be the superposition of $B1 = \{A1, A2, A3, \ldots\}$ or $C = \{B1, B2, B3, \ldots\}$;
- $\{Z_1, Z_2, Z_3, \ldots Z_n\}$ are the sub-feature vectors being combined for the individual nodes of Figure 1. Each Z_n itself can be a compound vector representing a sub-workflow or a complex vector description for an individual service step, built using the methods described in [11];
- $p_0, p_1, p_2, \ldots$ are a set of known *atomic* role vectors used to define the current position or step in the workflow.
- cx is the chunk size of vector Z_x, i.e., the number of sub-feature vectors being combined; and
- $StopVec$ is a role vector owned by each Z_x that enables it to detected when all of the steps in its (sub)workflow have been executed.

The key to understanding the power of eq. (1) as a hierarchical binding scheme results from the fact that the cyclic_shift operator distributes over XOR and we stipulate that cyclic_shift takes precedence over XOR. This means that if a compound vector, such has Z_x, as shown in eq. (2), is used as sub-feature vector for a higher level concept then the p vectors associated with each sub-feature vector Z_n are automatically promoted to a new value by the cyclic_shift.

In consideration of eq. (2) we can see that it is effectively a slot encoding scheme which are known to be brittle when comparing objects that are even slightly out of sync. This means that if we encoded two sentences, for example,

$$\begin{aligned} S1 &= p_0^0 \cdot My^1 + p_0^0 \cdot p_1^0 \cdot name's^2 + p_0^0 \cdot p_1^0 \cdot p_2^0 \cdot Jim^3 \\ S2 &= p_0^0 \cdot Hello,^1 + p_0^0 \cdot p_1^0 \cdot my^2 + p_0^0 \cdot p_1^0 \cdot p_2^0 \cdot name's^3 \\ &\quad + p_0^0 \cdot p_1^0 \cdot p_2^0 \cdot p_3^0 \cdot Jim^4 \end{aligned}$$

Then S_1 and S_2 would give a poor HD match because all but the first term is permuted differently and each term after *'The'* are orthogonal to their counter parts in the other sentence. Nevertheless, the scheme can prove surprisingly successful when the data is segmented frequently. For example if we

are encoding a book and we use a variable chunk size that matches the natural punctuation breaks then there are multiple opportunities to re-sync. The wording of a particular sentence may vary which would damage the comparison for such a sentence, however, the storey line is expected to follow and so subsequent sentences may be very similar. Where we wish to allow for variations in sentence structure we employ a local context scheme in conjunction with eq. (1) that we call *shaping*. This is simply a weighted binding method that adds an additional layer to the vector hierarchy where each new vector is a weighted sum of the surrounding vectors.

IV. The Hamlet Analogy

To illustrate how this concept could can be realized we use an example that has been described in detail in [11] using Shakespeares play Hamlet as a representation of a typical workflow. In this representation the various actors, who perform their part by saying the appropriate words at the correct time and place, are the analogue of software agents that are required to perform various sub-tasks in a workflow, i.e., an ordered sequence of tasks. To construct a vector representation of the entire play we begin by constructing a set of symbolic vectors that represent each letter of the alphabet plus the necessary punctuation. Each of the approx 4000 unique words can be constructed, using the vector binding and superposition operations such that there is a unique vector for each word. Using the word vectors each of the stanzas that an actor speaks can be similarly constructed into stanza vectors. The stanza vectors can in turn be bound together into scenes of the play and recursively these vectors can be bound into acts and finally into a vector that semantically represents the whole play, approximately 27000 words. In [11] the actors are used as an analogue of software agents that are distributed across a communications network and these agents operate together with other software agents that hold a representation of the vectors at the higher semantic levels. If the high level vector is injected into the network then it has been shown how this vector can be recursively unbound such that the entire play is performed in the correct order. In[11] it is also shown how more complex workflow configurations can be represented and executed in a distributed decentralized setting.

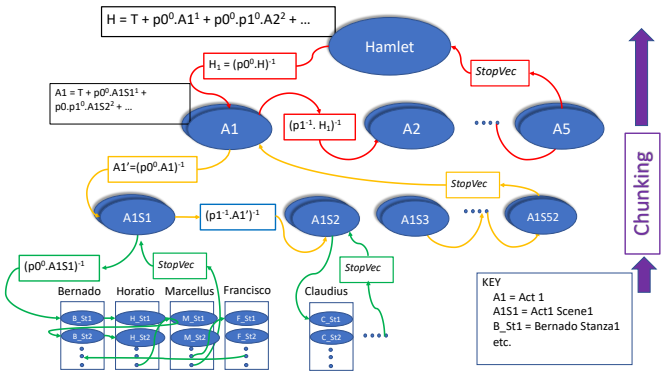

Fig. 1. Vector Chunk tree representing Hamlet, chunking proceeds from the bottom up

A. Communications Example

Each bubble in Figure 1 represents a symbolic vector which itself is constructed through a process of combining the vectors from the level below. Each of the vectors at the different levels representing the semantic meaning of all of the component vectors with a single vector at the top level semantically representing the entire play. To communicate the play the single top-level vector is sent to the coalition partner. In the Hamlet example the top level vector is composed by superposing the 5 vectors each representing the five acts of the play. Since the binary vector is a distributed representation it can be truncated and only the first 1000 bits need to be transmitted in order for the vector to be recoverable by the recipient. If the recipient has a copy of a version of the play and has constructed a 10 kbit vector using the recursive binding process, then by measuring the Hamming Distance between the received vector and their own vector representation, they can recognize that this is the 'Hamlet' vector with high probability. The recipient can then perform any task that such a vector represents (e.g. read out their own entire copy of the play) and no further communication is required.

If the recipient does not recognize the vector i.e. it is below the threshold Hamming Distance, then they can unbind the received vector which will give them noisy versions of the component vectors at the next lower semantic level (in this case the Acts of the Play). Performing the Hamming Distance comparison on these vectors may enable the recipient to recognize the vectors for Acts 1, 2, 4 and 5 but they do not recognize the other vector. Since the vectors are ordered the recipient might infer that this vector represent Acts 3 but they have no local representation of this part of the play. They cannot unbind this vector to determine its constituent vectors since it is a 'noisy' copy. To determine what the component vectors are the recipient needs a clean version of this vector. This can be obtained by requesting, either from the original party or from any other partner in the network, a clean copy of this vector. This can be achieved simply by a multi-cast of the noisy vector with a request for the clean version. Any partner within the coalition multi-cast group can provide the required information. On receipt of the clean version then this vector can be unbound to its component vectors and if these vectors are known then the details of the Act 3 vector can be determined. The process can be recursively repeated until all of the necessary information has been collected. This mechanism significantly reduces the communications bandwidth requirement, essentially by trading bandwidth for distributed knowledge, stored in symbolic vectors, across the coalition. Hence the concept of the 'distributed brain'.

One obvious question is, do the different parties have to have exactly the same version of the play to perform the matching? The answer to this question depends on the degree of semantic similarity that is required in order for the recipient to understand and act on the content of the received message. To illustrate how the semantic matching works we show in Figure 2, a comparison of the Hamming

Distance (HD), actually 1-HD so the value 1 represents a perfect correlation, between vectors generated from different editions of the Hamlet play for the vectors representing the entire play and for the vectors representing Acts 1 to 5.

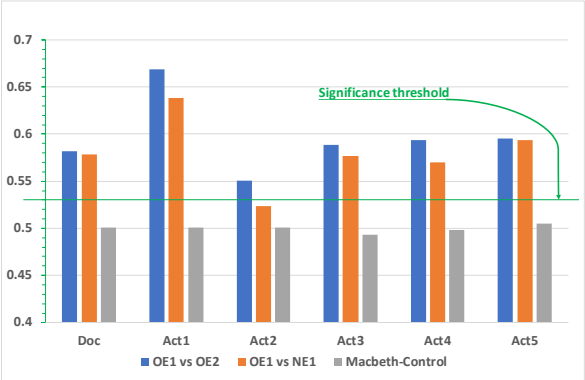

Fig. 2. Vector comparisons between different versions of Hamlet at the level of the entire play and for the individual Acts of the play expressed as 1-HD

The comparisons are made between two Old English (OE) versions of the play and comparisons between an OE and a modern New English (NE) version of the play. Also shown is a comparison with vectors generated from the play Macbeth. Comparison of two random vectors produces a Hamming Distance of 0.5 with high probability and so the vector comparison with the Macbeth vectors shows that these two plays are essentially uncorrelated (i.e. orthogonal). If the vector comparison is above the significance threshold of 0.53 then the two vectors are highly correlated and are semantically the same with high probability. The results show that at the level of the entire play and for all acts the two OE versions are semantically the same. Perhaps, surprisingly, the comparison between the OE and NE versions also shows that these are also semantically similar except for Act 2 which does not match. It is possible to ensure that the a match only occurs if the recipient has an exact match. Details of how to determine appropriate thresholds to ensure that the degree of semantic match is sufficient is beyond the scope of this paper.

Similar comparisons can also be made at the lower semantic levels and Figure 3 shows the equivalent comparisons at the level of the scenes of the play. At the scene level the OE1 vs OE2 are all semantically similar. The OE vs NE comparison is below the threshold for Act2 Scene1, Act4 Scene7 and is marginal for Act3 Scene 1 but otherwise the versions are still semantically similar.

B. Command and Control Example

One of the key questions for coalition operations is how do we perform coordinated missions that require multiple inter-related tasks to be performed by coalition partners that can perform the required tasks but might do so in a slightly different way to the way our own forces would perform the task. In unilateral, single nation operations, commanders can issue a command to perform a mission and, because there is a high degree of shared understanding, then the various units

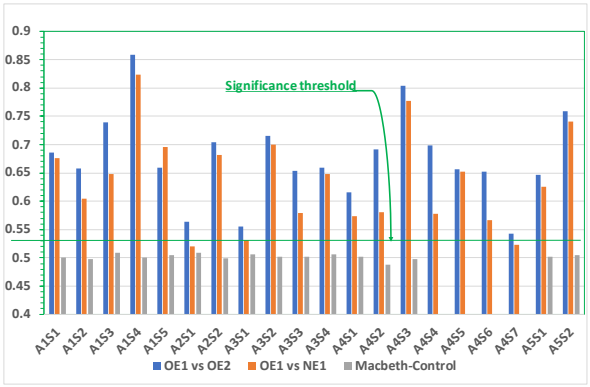

Fig. 3. Vector comparisons between different versions of Hamlet at the level of the individual Scenes of the play expressed as 1-HD

that are going to perform the task know precisely what to do in response. This is a direct consequence of common training and the rehearsal of operations. In coalition operations the command Execute Plan B may not be at all understood by units that could contribute to achieving the desired goal but if the message could in some sense convey all of the details of the plan in a way that could be subsequently unambiguously interpreted then this would offer a significant advantage. It is our contention that the use of symbolic vector representations directly facilitate this capability.

The analogy between our Hamlet play example and the command and control actions required to direct the operations is therefore not difficult to make. Indeed, the terminology used such as theatre of operations, players, moves, movements, actors, opening moves, finales etc. makes the analogy compelling. The plan is the equivalent of the whole play and the actors are the units and resources who will implement the plan. Issuing the command is the equivalent of transmitting the high level vector. The equivalent of the Acts and Scenes represent the command structure of the forces which by way of example could be: Company (Acts); Platoon (Scenes); Squad (Actors); with the Fireteam's instructions being the equivalent of the stanzas. The commander does not necessarily need to know which of the available resources will actually perform but can be assured that resources best placed to perform the sub-tasks will be selected. To do this they need to have a clear understanding of the sequence of sub-instructions that need to be performed by units at the next level and can issue appropriate instructions. Knowing precisely which actions to perform often depends on the context of the overall operation and therefore if the command can in some sense maintain the mission context then this assists in a correct interpretation of the higher level command. This is precisely what the symbolic vectors provide. The vector is not simply a single command but essentially embeds a description of the entire operation that the commander is requesting to be performed. As the operation proceeds the vectors are recursively unbound by the collaborating units and although there may be some variation

in how the sub-tasks are performed the overall mission will be correctly performed.

The vectors have the property that they can then be unbound to unravel the next step in the mission which is again injected into the communications network and the next action or set of actions is performed by units that match the next vector. Details of how the process works in an unreliable wireless network environment such as a Mobile Ad-Hoc Network (MANET) are presented in [11].

This new approach to C2 offers significant advantages that will be essential for future coalition military operations.

V. Conclusions And Future Work

In this paper we have shown how binary symbolic vector representations can be used to address a number of the key inter-operability issues that impact on future C3, particularly in distributed decentralized coalition operations. We have shown that such representations can be used to efficiently communicate information between coalition partners by leveraging shared semantic understanding to minimize communications bandwidth whilst avoiding ambiguity and the possibility of misunderstanding. We have also shown how such representations can be used to communicate and co-ordinate command and control functions and how units that are distributed across a theatre of operations know which actions to perform by exchanging the symbolic vectors that semantically describe the mission tasks. Symbolic vector representations can significantly reduce the communication bandwidth required to perform these tasks. Whilst we recognize that the 'Hamlet' example is not a perfect analogy of a complex military operation it serves to illustrate the power of the approach. We are currently working on a number of military vignettes that will better illustrate how the concept applies to typical coalition operations and these will be reported in a future paper.

We are currently investigating how to extend the binary symbolic vector representation to cover cases where completely different words/symbols may be used to convey the same semantic information. Various techniques have been developed for learning semantic vector representations from a large text corpus e.g. Word2Vec [2] and we are investigating how these vector spaces can be mapped into the binary vector representations.We are also investigating how symbolic vectors can be used to represent and share deep neural network models between coalition partners. We believe that the types of symbolic vector representations described in this paper will make a significant contribution toward the vision of future C3I as a 'distributed federated brain'.

Acknowledgements

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. 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 U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] C. Eliasmith. *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.

[2] Y. Goldberg and O. Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[3] G. E. Hinton. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75, 1990.

[4] M. N. Jones and D. J. K. Mewhort. Representing word meaning and order information in a composite holographic lexicon. *psychological Review*, 114(1):1–37, 2007.

[5] P. Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.

[6] D. Kleyko. *Pattern Recognition with Vector Symbolic Architectures*. PhD thesis, Luleå tekniska universitet, 2016.

[7] T. A. Plate. *Distributed representations and nested compositional structure*. University of Toronto, Department of Computer Science, 1994.

[8] T. A. Plate. *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. CSLI Publications, Stanford, CA, USA, 2003.

[9] G. Recchia, M. Sahlgren, P. Kanerva, and M. N. Jones. Encoding sequential information in semantic space models: comparing holographic reduced representation and random permutation. *Computational intelligence and neuroscience*, 2015:58, 2015.

[10] C. Simpkin, I. Taylor, G. Bent, G. De Mel, and S. Rallapalli. https://dais-ita.org/sites/default/files/dais-2017-paper-final.pdf. In *DAIS 2017 - Workshop on Distributed Analytics InfraStructure and Algorithms for Multi-Organization Federations, part of IEEE Smart World Congress 2017.*, 2017. https://dais-ita.org/sites/default/files/dais-2017-paper-final.pdf.

[11] C. Simpkin, I. Taylor, G. A. Bent, G. De Mel, and S. Rallapalli. Decentralized microservice workflows for coalition environments. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2017.

[12] D. Verma, G. Bent, and I. Taylor. Towards a distributed federated brain architecture using cognitive iot devices. In *9th International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE 17)*, 2017.