USING CONCEPTUAL SPACE AND CULTURAL EVOLUTION IN LANGUAGE GAME

Theerapol Limsatta

A Dissertation Submitted in Partial
Fulfillment of the Requirements for the Degree of
Doctor of Philosophy (Computer Science and Information Systems)
School of Applied Statistics
National Institute of Development Administration
2016

USING CONCEPTUAL SPACE AND CULTURAL EVOLUTION IN LANGUAGE GAME

Theerapol Limsatta

School of Applied Statistics

Associate Professor [illegible] Major Advisor
(Ohm Somil, Ph.D.)

The Examining Committee Approved This Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science and Information Systems).

Associate Professor [illegible] Committee Chairperson
(Surapong Auwatanamongkol, Ph.D.)

Associate Professor [illegible] Committee
(Ohm Somil, Ph.D.)

Associate Professor [illegible] Committee
(Pipat Hiranvanichakorn, Ph.D.)

Assistant Professor [illegible] Committee
(Rawiwan Tenissara, Ph.D.)

Assistant Professor [illegible] Dean
(Sutep Tongngam, Ph.D.)

March 2017

ABSTRACT

Title of Dissertation	Using Conceptual Space and Cultural Evolution in Language Game
Author	Mr. Theerapol Limsatta
Degree	Doctor of Philosophy (Computer Science and Information Systems)
Year	2016

Agreement on word-object pairing in communication depends on the intensity of belief that gradually emerges in a society of agents under the condition that no one is born with embedded knowledge. In know-nothing word-object pairing, the agents in communication find meaning until they reach a consensus on what an object should be called. A language game is a social process of finding agreement on word-object pairing which enables its communication in a multi-agent system. In this research, techniques are proposed to discover the association between a word and agents' beliefs on an object using self-organizing maps and flexible searching with a cultural algorithm to find the meaning for concepts in a space that retains the agents' beliefs in three quality dimensions represented by colors: red, green, and blue. The techniques were evaluated in a variety of scenarios using four significant measures: coherence, specificity, success rate, and word count. The results show that social agents were able to quickly reach mutual agreement for multiple listeners by searching their social beliefs and using a cultural algorithm to search for cultural beliefs.

ACKNOWLEDGEMENTS

This research started as an interest in Wittgenstein's philosophy, principally language games, since the application of his philosophy is very new in computer research, especially in Thailand, and it is possible that this research using language games is the first of its kind in Thailand. There have been many times when I discussed how to apply language games in computer science and information systems, and there were many questions, some of which were difficult to clearly explain. However, my advisor (Assoc. Prof. Dr. Ohm Sornil), encouraged me and gave very valuable comments over many years. I greatly appreciate and respect his patience and guidance, and I fully intend to follow his teaching guidelines.

I would also like to thank the committee members (Assoc. Prof. Dr. Surapong Auwatanamongkol, Assoc. Prof. Dr. Pipat Hiranvanichakorn, and Asst. Prof. Dr. Rawiwan Tenissara) for their comments and questions to help clarify my understanding and make a reasonable solution in this study.

Theerapol Limsatta
February 2017

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figures **Page**

ABBREVIATIONS

Abbreviations	**Equivalence**
BMU	Best-matching Unit
CA	Cultural Algorithm
COH	Coherence
CS	Conceptual Space
EC	Evolutionary Computation
SPEC	Specificity
SOM	Self-organizing Map

CHAPTER 1

INTRODUCTION

1.1 Motivation

The meaning of an object is represented in the world by humans using language symbols, but in relation to what? Humans use meaning to identify an object in the world for communication purposes. Each community has the same meaning for the same object, but they represent it in language symbols in different ways. There must be some rules for each community related to their use of symbols to find meaning, and this phenomenon has been investigated since the early twentieth century by both Ferdinand de Saussure (a Swiss linguist (1857-1913)) and Charles Sanders Pierce (an American logician (1839-1914)), both helping to lay down the foundations of semiology. Their ideas had a significant effect to Ludwig Wittgenstein (known primarily as a philosopher in logic), not directly but by being partly communicated to him via Frank Ramsey (Pietarinen, 2003).

Wittgenstein defined a language game, i.e. "rule, "use", and "play with rule", in his writings named "Philosophical Investigations" from the mid-1930s. It inspired many of researchers to find the origin of language that humans use with less ambiguity and variability for successful communication under the belief that if we discover the origin of language, it will also help to discover two important things: first, the origin of the mind (referred to as part of artificial life), and second, it can support computation linguists with two basic problems: ambiguity and variability, known as polysemy (a related sense or meaning), homonymy (an unrelated sense of meaning), and synonymy. For example, Stuber, Hassas and Mille (2006) showed that a computer agent helped an actor (human) to fulfil her task by providing contextual access to her individual experience. To do so, some meanings on simulation in a language game needed to be shared by the actor and her assistant (the computer agent) to allow their mutual understanding during this interpretation of the current trace. Moreover, Steels (2010) reported that robots (so-called autonomous agents) can communicate not only single but multiple words using case grammar, which is a kind of linguistic syntax emphasizing on word function.

They are more uses, as can be seen in a number of papers in literature review chapter, especially on the idea of conceptual space presented by Gärdenfors (2000), and also Logan's (2004) recommendation that a time dimension could be used instead of a space dimension to form a dynamic cognitive system, namely, the human mind. In the article by Loreto, Baronchelli and Puglisi (2010), they indicated that cultural negotiation is needed for a hierarchical level: a superior linguistic structure on top of the individual-dependent discrimination layer. When concepts come from the perception of the physical world mapped by language in some spaces of the brain plus the time dimension, then time in a language game serves in this way, as found in some parts of the physical dimension space suggested by Lindh-Knuutila, Honkela and Lagus (2006) and Beule and Bleys (2010), and some parts of the time dimension space suggested by Lenaerts, Jansen, Tuyls and Vylder (2005).

1.2 The Research Objective

There is little research on conceptual space implemented for multiple listeners that includes a time dimension, such as cultural belief. Furthermore, there are also few studies on language games with multiple listeners. Even in the research of Li, Chen and Chow (2013), there is still no coverage of conceptual space.

The aim of this research is to use conceptual space, the time dimension, and multiple listeners in language games to better develop agent communication by improvements in ambiguity and variability. The research is extended using conceptual space with flexible searching in a self-organizing map (SOM) to construct a similar conceptual space for supporting improvements in ambiguity using coherence and specificity measures. The time dimension is applied along with a cultural algorithm to search for global belief, in which all agents hold the same belief, using long runs. Additionally, multiple listeners in a cooperative concept are included to support an improvement of variability by measuring communication success and lexical item size.

1.3 Concept of Meaning

1.3.1 Mental Representation

Meaning from a mental standpoint is a form of representation, but what is the form? The traditional theory especially refers to picture theory when it refers to an object name (Wittgenstein, 2001). However, this theory has one drawback: is the picture one individual refers to the same object in another's mind because both may have experienced it in a different way? For example, for a common noun like triangle, one speaker may have a mental image of an equilateral triangle whereas another's might be an isosceles or scalene triangle. To relieve this problem, Wittgenstein suggested that an individual's mind has a picture of resemblance. He said, "We see a complicated network of similarities overlapping and criss-crossing" (Wittgenstein, 1963: 66). He compared the relationships between different games to the family resemblance that exists in the appearances of members of the same family. As seen in comparing the game Solitaire or a TV game show, for example, we know that they belong to a category of game but they might not be defined by any core of shared attributes. Even if using a list of necessary and sufficient conditions, the core shared attributes to classify them as the same category may not be found. However, family resemblance is a certain set of possible attributes which tie the members to the same category.

1.3.2 Prototype Theory

Rosch and her colleagues, a group of psychologists at the University of California at Berkeley, generalized the family resemblance structure of Wittgenstein. This group conducted experiments in order to test the category in which subjects were asked to consider examples from different categories, like BIRD, VEGETABLE, FURNITURE, and CLOTHING, and rate them on a seven-point Likert scale for each category: "1" meant that is was considered an excellent example, "4" indicated a moderate fit, whereas "7" suggested that it was a very poor example. The results showed that agreement was high for the items rated as very good examples of a particular category. For example, a robin was the best example of a bird, a pea the best example of a vegetable, and a chair the best example of furniture. In accordance with these examples, Figure 1.1 shows birdiness rankings from Rosch's testing.

To summarize, Rosch's work indicates that people seem to have some idea of the characteristics of exemplars to categorize common objects. The decision on category is not an exact match; it just has to be sufficiently similar, but not necessarily an exact match. This is a so-called prototype in Rosch's work and family resemblance in Wittgenstein's work.

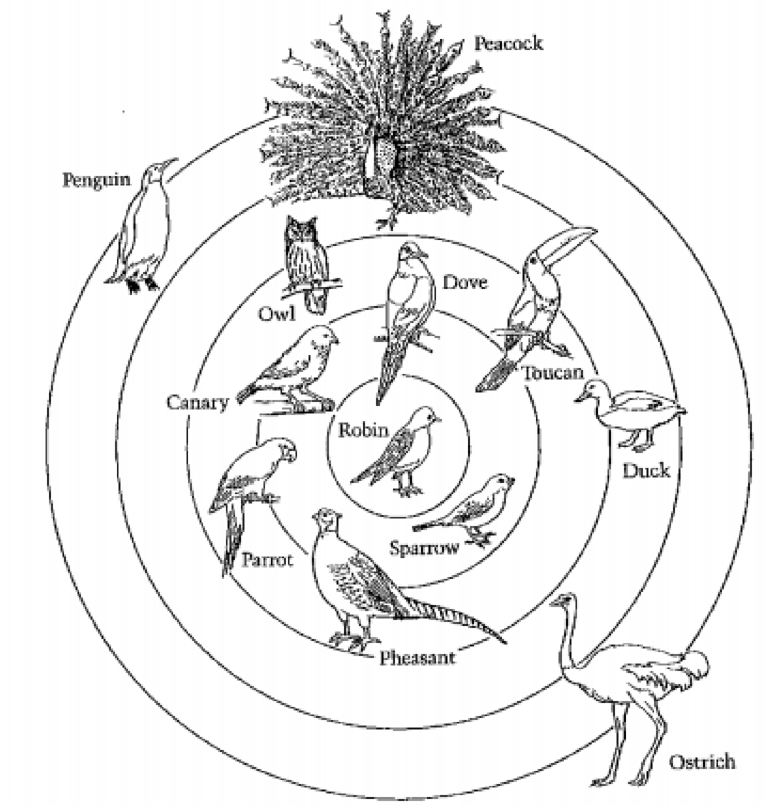


Figure 1.1 Birdiness Rankings
Source: Aitchison, 2012: 54.

1.3.3 The Idealized Cognitive Model

Prototype models have fuzzy boundaries and category members do not all share the same discrete features. These ideas form a type of complex classification called radial categories that show the typicality effect, which comes from a culture where the individuals hold a social belief together. Lakoff (1987) proposed the idealized cognitive model theory (ICM) as a prototype for an idealized cognition of social beliefs. For example, the word 'bachelor' means simply "an unmarried man", but in some culture it means bachelorhood, and ‘marriage’ (a frame or ICM) is a monogamous union between eligible people, typically involving romantic love. In the case of the former word, an idealized model about the word ‘bachelor’ forms from general knowledge due to its use. Later, this theory plays an important role in the implementation of the cultural algorithm.

1.3.4 Conceptual Space

A mind has something like an idea to represent objects in the world that as a prototype can be mapped by symbols or word to form meaningful expression. The symbols represent language propositions of various kinds that equate to logical relationships, so the mind processing constructs a logical sentence to show its meaning. This representation has two levels: one is the meaning in the mind and the other is translating the meaning into symbols. This cooperation on two levels is a sentential paradigm forming an implicit methodology for much of the research into AI (Gärdenfors, 1996). The combination of symbols to form expression in a language of thought is referred to as mentalese by the author, who claimed that if mentalese processing is likened to computer processing, then the first processing level is mind mapping of a thought response to an object in the world and the second level is translation of thought to express it in language form, which constitutes metaphorical expression between computer and mind processing.

Sensors and actuators can also be metaphorical in relation to the two levels of interconnection in mentalese processing. The sensors accept physical data like weight, velocity, etc., while the actuators translate the physical data to make changes driven by voltage. However, in the mentalese process, one of the two steps must involve symbolism of thought that is not specific to any natural language and the other ensures the application of the natural language of a specific community. Furthermore, Gärdenfors claimed that in all sentential paradigms, the first step of mental presentation cannot be reduced to neurobiological or other naturalistic categories, so it must be something that works similarly to a pattern that is a percept.

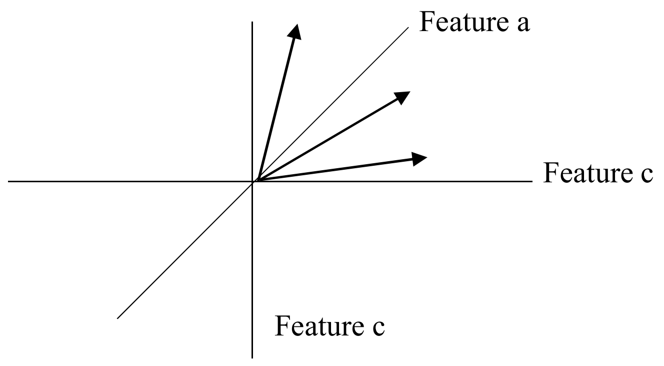


Figure 1.2 Dimensions of Thought in Conceptual Space

For possible simulation in a computer, the geometric conceptual space proposed by Gärdenfors is the best one at this time. In brief, space conceptualized as a geometric structure is represented as a quality of dimensions. Each dimension represents a world perception or feature and the dimensions form the geometric space as a category of the perception that represents an example of a prototype, which he later referred to as the best example of a prototype.

1.3.5 Linguistic Words

To form meaning, some lexemes or semantic words are used to explain conceptual representation and the relationship between concepts, as has been stated for conceptual networks. In linguistics, a lexeme is represented as a lexical field, which to be clear, is also definded as a lexical relation as follows.

Homonymy has unrelated sense or meaning whereby an identical word can reference a different concept. A homonym is divided in two types: homographs (the same written word), and the homophones (the same sounds in a spoken word). For example, the word "bank" has two meanings: one for financial institution and the other is the land next to a river.

Polysemy is where a sign (e.g. word, phrase, or symbol) has multiple meanings. For example, the word "hook" in Collins English Dictionary has several distinct but similar meanings like 1. a piece of material, usually metal, curved or bent and used to suspend, catch, hold, or put something. 2. short for fish-hook. 3. a trap or snare, and also more esoteric meanings, such as “a thing designed to catch people’s attention” when referred to in a sales context. However, it is important to note that although homonymy is accidental, polysemy is not.

Synonymy is used to explain different phonological words which have the same or very similar meaning, e.g. large and big are synonyms but are used in different situations and with different collocations. It is the opposite of polysemy and homonymy, where the same word has a different meaning. Synonymy, homonymy, and polysemy are the ambiguity and variability of the most basic and pervasive phenomena characterizing lexical semantics.

A lexeme is a basic word of a language and may consist of one word or several words to form a particular meaning., and a lexicon is an inventory of the lexemes of a particular language. An English dictionary is an example of a lexicon of the English language.

1.4 Language Game

To model conceptual space in this study, a simulate language game based originally on the notation of Wittgenstein (1963) is used. To demonstrate this, consider an example dialogue taken from a language game reported by Loreto et al. (2010) between two agents, a speaker and a listener, within a particular contextual setting; their language game considered a question about opinion where the meaning as a judgment does not only depend on “I think” but on the question, “Don’t you think so?”.

1.4.1 A Simple Language Game

Loreto et al. (2010) listed the simplest rules of the language game process as follows:

1) The speaker randomly selects a topic from the set of topics on the current context.

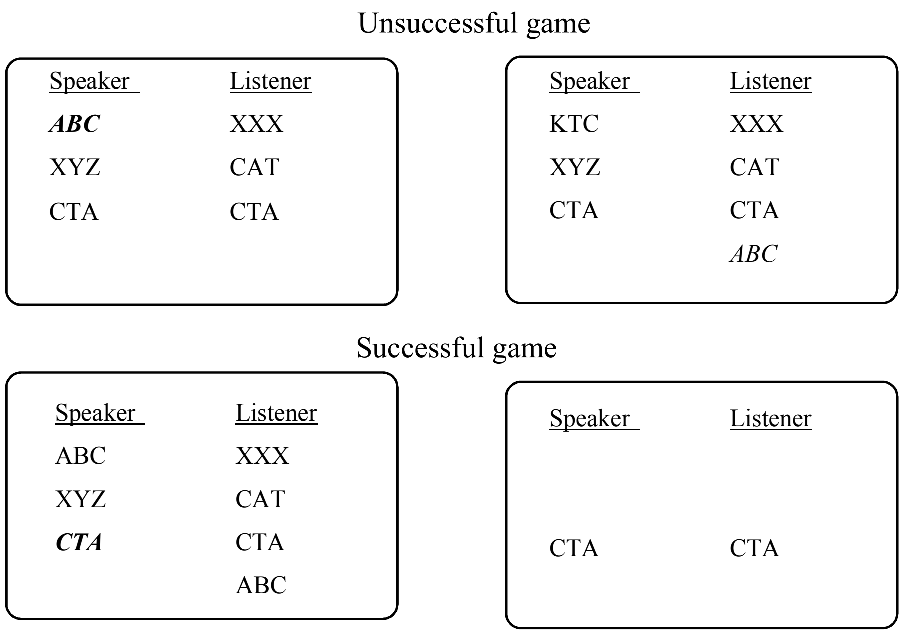
2) The speaker retrieves a word from its inventory associated with its topic, or, if its inventory is empty, the speaker invents a new word.

3) The speaker utters the selected word to the listener.

4) The listener searches the word named by the speaker in its inventory and that word is associated with the selected topic.

5) If the listener finds that word, it accepts that word. Thus, the interaction is a success and both players maintain their inventories only with the successful word and delete all the others.

6) If the listener does not find the word associated with the selected topic and named by the speaker in its inventory or the word is associated with a different topic, the interaction is unsuccessful and the listener updates its inventory by adding an association between the new word and the selected topic.



Unsuccessful game

Speaker	Listener
ABC	XXX
XYZ	CAT
CTA	CTA

Speaker	Listener
KTC	XXX
XYZ	CAT
CTA	CTA
	ABC

Successful game

Speaker	Listener
ABC	XXX
XYZ	CAT
CTA	CTA
	ABC

Speaker	Listener
CTA	CTA

Figure 1.3 Simple Language Game Examples

Figure 1.3 is an illustrated example of simple language games. In the unsuccessful game, the speaker selects the italicized word (*ABC*) in the top left figure. The listener does not possess that word (*ABC*), so the speaker adds that word to its inventory (top right of the figure). On the other hand, in the successful game, the speaker utters the 'CTA' word (bottom left of the figure) to the listener and the listener finds that word in its inventory, so the listener accepts that word and both listener and speaker erase their inventories except for the accepted word (bottom right).

1.4.2 Category Game

For higher forms of agreement, the so-called category game focuses on the process by which a population of individuals manage to categorize a single perceptually continuous channel. Loreto et al. (2010) and described the sequence of a category language game (depicted in Figure 1.4) as follows:

In the case of game 1:

1) In the game, two players, a speaker and a listener, are randomly selected from the population.

2) Two topics are randomly presented to the two players and the speaker first selects the topic ("a" in this example).

3) The speaker has to discriminate the chosen topic ("a") by creating a new boundary in the speaker's rightmost perceptual category at position (a+b)/2.

4) The two new categories inherit the word inventory of the parent perceptual category (the words "green" and "olive" in this example) along with two different new words ("brown" and "blue").

5) Later, the speaker retrieves the words associated with the perceptual category that contains the topic.

6) There are two possibilities. If a previous successful communication has occurred with this perceptual category, the last winning word is chosen, else the last created word is selected.

In this example, the speaker chooses the word "brown" and utters that word to the listener. The result of this game is a failure because the listener does not have the word "brown" in its inventory. The speaker shows the topic in a non-linguistic way (e.g. by ostensive definition or pointing at the topic), then the listener adds the new word to its inventory.

In game 2:

1) The speaker chooses topic "a", finds the topic that was already discriminated, and utters the last successful communicated word (the word "green" in this example).

2) Both the speaker and the listener know this word, and so they define the topic correctly. This is a successful game because the speaker and the listener eliminate all competing words (the word "green" from the perceptual category in this example).

In general, when ambiguities exist, the listener finds the word associated with more than one category containing and topic, but these ambiguities may be resolved by ordering the words.

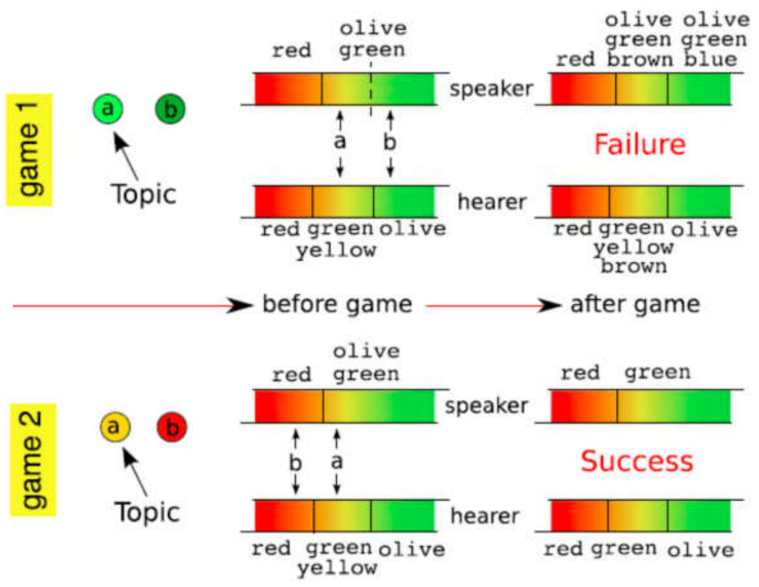


Figure 1.4 Category Language Game Examples

Source: Loreto et al., 2010: 273.

1.4.3 Level of Social learning in a Language Game

Horizontal, vertical, oblique, and orthogonal transmission are defined levels of learning. Horizontal transmission focuses on one generic generation communication within a population, while vertical transmission is the communication between different generations, like a parent and child. Oblique transmission addresses communication between different role-relationships within a culture and also different cultures. Orthogonal transmission is a special case because it combines horizontal and oblique transmission to solve conflicts caused by genetic and cultural differences.

The acquisition framework is a major form of cultural transmission. Gong (2010) created the framework for cultural transmission with three levels (horizontal, vertical, and oblique) as shown in Figure 1.5.

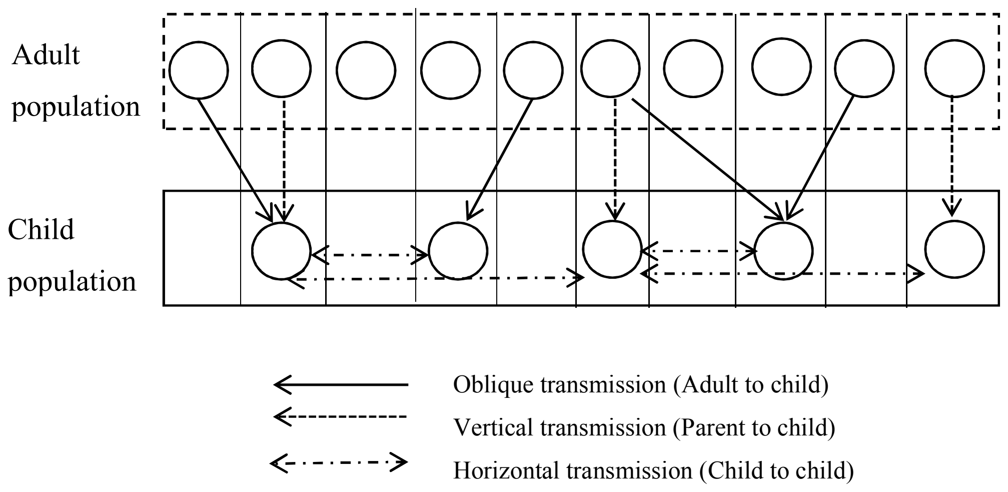

Adult population
Child population
Oblique transmission (Adult to child)
Vertical transmission (Parent to child)
Horizontal transmission (Child to child)

Figure 1.5 Cultural Transmission with 3 Levels
Source: Gong, 2010: 3.

1.4.4 Types of Language Game

There are several types of simulated language games compartmentalized into three categories according to Lindh-Knuutila et al. (2006) as follows:

1) Observational games: the speaker and listener know in advance the topic of the game. The listener learns the name the speaker uses for that topic, whereby learning is associative. For the proposal in this study, categorization is mostly similar to an observational game.

2) Guessing game: the players (a speaker and a listener) are presented with a small number of topics. The listener must guess which topic is the one the speaker is referring to. However, at the end of game, the speaker gives corrective feedback to the listener as to whether the guessed word was right or not.

3) Selfish game: the listener does not receive any feedback on words from the speaker, thus the listener must infer the meanings of words from their co-occurrences in different contexts or situations. The game is called ‘selfish’ because the speaker does not care whether the listener clearly understands or not.

CHAPTER 2

LITERATURE REVIEW

There are many related approaches to language games, but researchers have rarely investigated the concept space with many listeners using SOMs and including a time dimension with a cultural algorithm. In general, language game searches for cooperative cognition in a society to do with its community, and so researchers have presented many inter-disciplinary methods to study this. However, the main functions in language games can be used to form new concepts with the help of topics associated with them.

2.1 Forming a Concept

2.1.1 Using Conceptual Space

Since Gärdenfors (2000) introduced conceptual space based on geometric form, many studies such as the one by Chella, Frixione and Gaglio (2001) have shown how to implement conceptual spaces for the computer vision-based idea of Gärdenfors by defining conceptual semantics of the symbols-grounded data from sensors. Gärdenfors defined three levels of conceptual representation: sub-conceptual space, conceptual space, and symbolic. The data objects from cameras consist of complex data with many qualities reduced to a simple form for the presentation of sub-conceptual space, which then becomes conceptual space with a set of convex concepts to represent the semantic region mapped to symbolic words. Figure 2.1 shows the relationship between the three level of Gärdenfors' idea and the general architecture of computer vision. To map meaning to symbolic words, the semantic regions in the internal structure of concepts based on a geometric dimensional map is converted to 3D primitive shapes and also labelled with words. The basic shapes are defined by Mortenson (1997) using Constructive Solid Geometry (CSG) that later became widely used in computer graphics. CSG representation is based on super-quadrics that are geometric shapes according to Gärdenfors' concept.

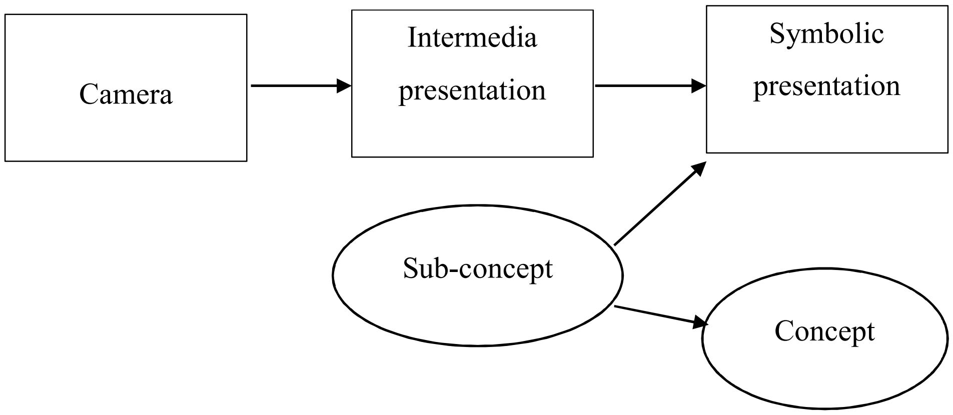


Figure 2.1 Mapping the Three Levels of Concept for Computer Vision Architecture

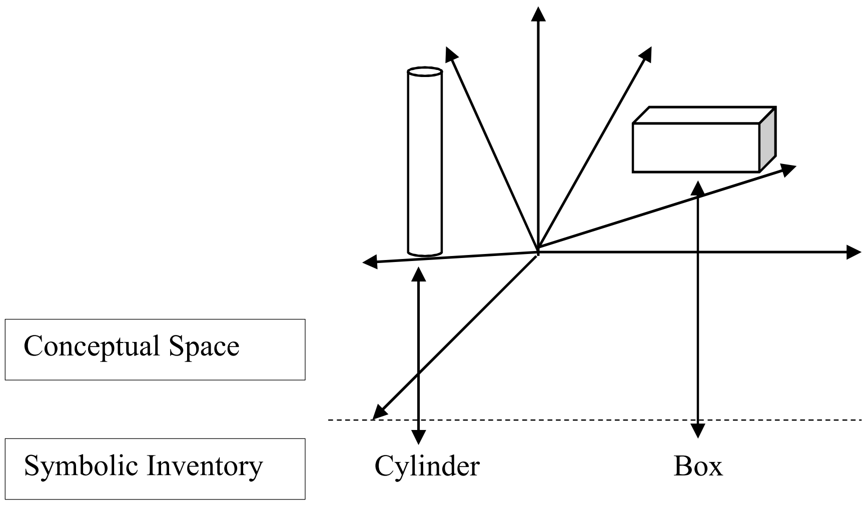


Figure 2.2 Mapping of 3D Primitive Shapes to a Symbolic Inventory

A SOM is another structure that can be implemented using conceptual space, as suggested by Lindh-Knuutila et al. (2006) who investigated conceptual space using SOMs with three color dimensions (red, green, and blue) and using the social dimension as a rule for an observation game to form constraints on conceptual space that can then be considered as part of a symbolic level. Objects that are played in the language game are colors that are mapped to the best map unit in the SOM structure and later labeled with words. However, in the gameplay, patterned structures in the SOM must be trained with eight different prototypical colors as if it were a baby within who the brain is not yet fully developed. The reasons why they used colors as objects for topics played in the language game are that object color observation and storage unit have a similar structure; they have the same three dimension features (red, green, and blue), and therefore communication is established easily. Moreover, this is an example that shows how success leads to meaning in environments of language game simulations. However, this simulation lacked another dimension that serves to improve the quality of conceptual space and should also be taken into consideration: the time dimension as a cultural constraint.

An SOM has many nodes that are memorized objects mapped to a region of its structure: so-called object belief. The mapping pairs have beliefs about objects that are updated from players' experiences in a language game. Figure 2.3 shows lexemes mapped to nodes in an SOM structure in an experimental language game reported by Lindh-Knuutila et al. (2006). As seen on this figure, many of the lexemes contain instances of homonymy, polysemy, and/or synonymy.

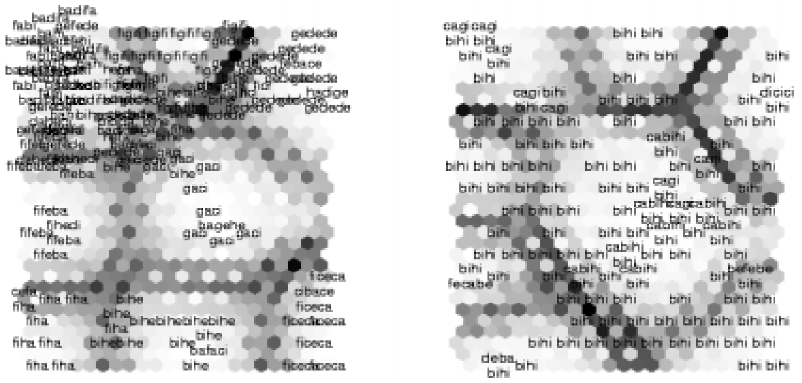


Figure 2.3 Two Example Conceptual Maps Implemented with SOM
Source: Lindh-Knuutila et al., 2006: 177.

Beule and Bleys (2010) develop a model showing that language concerning color terms freely change from the Old English period (c. 600-1150) to the Middle

English period (c. 1150-1500) without any extra level of competition and selection between two explicitly represented and predefined language strategies. To create models, they constructed a similar conceptual space but defined perceptual space using the standard Euclidian distance in color space. The perceptual and the conceptual spaces comprise a geometric structure, so it may be claimed that perceptual space is another form of conceptual space. Each player perceives colors as points in its private perceptual color space and its linguistic inventory. All example colors linked to the same term are said to form a linguistic category, so this technique is similar to the category language game described in chapter 1.

A statistical method such as principal component analysis (PCA) is a way to reduce the dimensionality of a dataset in the processing of sub-conceptual space layer (recall that this layer is the internal process to form a concept). McGovern, Lawry and Leonards (2014) used this technique to model conceptual space by focusing on vagueness. They defined conceptual space with two properties: the prototype, which is object knowledge defined within the conceptual space; and the threshold, which is a random variable representing an agent's estimate of an object's vagueness using the radius of the region corresponding to conceptual space. As can be seen in Figure 2.4, the concept is defined by a prototype P and a threshold ε, whose size is obtained by probability.

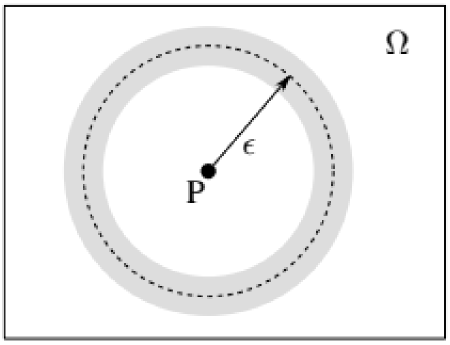


Figure 2.4 The Representation of a Concept in 2-Dimension Conceptual Space Ω
Source: McGovern et al., 2014: 2.

Before forming conceptual space, PCA can be applied to analyze the data from, for instance, an artificial fingertip (see Figure 2.5) to construct a viable feature space. The data on the features of a cloth's texture must be ignored as it is the least significant, i.e. the dimension along which there is minimal variance. Hence, the significant data is then the quality of the dimension formed conceptual space.

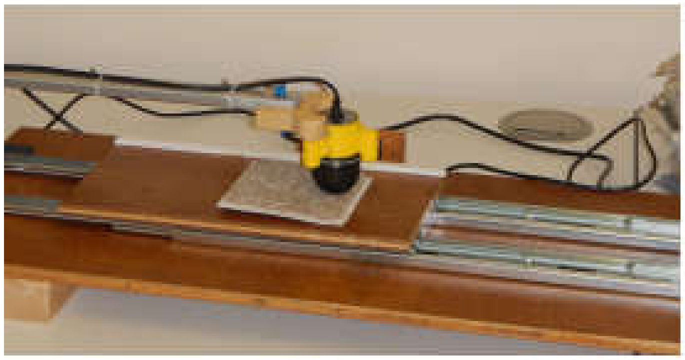


Figure 2.5 An Artificial Fingertip Reading Features on Texture
Source: McGovern et al., 2014: 2.

2.1.2 Non-use of Conceptual Space

There a number of non-uses of conceptual space, and all of them can form concepts by using graph theory.

Using a network of lexemes to form a concept can be found in the study of Lipowska and Lipowiski (2012). Nodes are linked within an adaptive weighted network, in which each agent has a lexicon acting as linked nodes to form a network of semantics. This means that a concept contains at least one network, and there can be many networks for each concept. However, in the process of learning words from communications that succeed or fail to readjust a weighted link, a pair of successful communications between agents is selected by

$$P_{ij} = \frac{w_{ij}}{\sum_{k=1}^{N} w_{ik}},$$

where the weights are

$$w_{ij} = \begin{cases} s_{ij} + \in for\ i\ \neq j \\ 0 \qquad for\ \ i = j \end{cases},$$

for i, j = 1, 2, …, N. The (positive and typically small) parameter ∈ ensures that a speaker can sometimes play the language game with agents with which its up-to-now communicative success rate is very small or even zero.

When using this network, each pair of the successful communications is memorized and kept as historical data, so the next time the successful word can be picked. Figure 2.4 shows the final stage for modeling a concept based on this network.

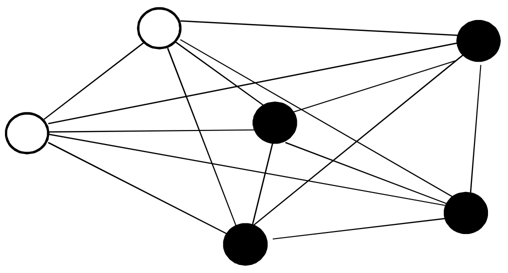


Figure 2.6 An Adaptive Weighted Network in the Final Stage of the Coarsening Process

The steps played in the language game in this model are as follows: (i) a black speaker (a black circle) selects a white listener (a white circle), which results in failure, and the white listener adds the word used by the black speaker to its lexicon; (ii) the white agent selected as a speaker utters the word to the black listener and the word it chooses to communicate is the one acquired in step (i), so this second step is successful.

The matrix for a lexicon is similar to the network but has the different way of updating the object-word pairing. Forming the matrix found in the simulation of Lenaerts et al. (2005), the following formula for creating a lexicon shows a number of words mapped with their meanings:

Lexicon = { (d_1, w_1), (d_2, w_2),…,(dj,w_2), …,(d_k, w_l), ..}, where d is the meaning of a particular object, and w is the word mapped to meaning d. They fixed the number of meanings and words to keep things simple, even though unrealistic in practice. The word-meaning associations can be used in matrix n = (d x w), and associated with value v: a specifier of the strength of association between a particular meaning and word having a range of 0 to 1. Hence, a value close to 1 means a strong relationship and one close to 0 means the opposite. With this representation, the lexical matrix behaves like an adaptive weighted network. The specifiers can also be represent in matrix form as (Lenaerts, et al., 2005: 568)

$$L^i = \begin{pmatrix} v_{11}^i & \cdots & v_{1w}^i \\ \vdots & \ddots & \vdots \\ v_{d1}^i & \cdots & v_{dw}^i \end{pmatrix}$$

The BA scale-free network found in Guo, Meng and Liu (2014) is a kind of network topology used to implement the formation of concepts in a language game. At each step in this network, a new node with *m* edges is preferentially attached to nodes in the existing network. The process is repeated until there are *N* nodes that avoid reconnection and self-connection, so the average degree of the BA scale-free network is approximately equal to 2*m*, and the minimal language game that evolved is as follows:

1) At the first step, the speaker *i* is selected from a random node and proposes an opinion *z* with probability $P_z = \frac{F_i^z}{\sum_v F_i^v}$, where v is the sum of all opinions in node *i* and *F* is the freshness.

2) The listener is selected from j neighbors, and if the jth listener has an opinion z, the negotiation is successful, the freshness of opinion z is renewed as $F_j^z = BF_j^z + (1-\beta)$, where $\beta = \frac{1}{\propto}$ (α is the personal ability to accept a new opinion), and the other opinions *u* in the memory of *j* are revised as $F_j^z = BF_j^u$. Otherwise, the game fails and adds opinion *z* to the memory of *j*.

3) After the negotiation has finished, the listener checks all opinions in the memory of j. If the opinion freshness goes to 0, the opinion and its freshness will both be dropped.

The key evolution of this game is that the model can reflect the common behavior in agents' (node) memories. For $\alpha > 2$, so $\beta < 0.5$ and the freshness of the latest transmitted opinion will increase. Note that in the language, there is no mention of topics or objects with meaning.

2.2 Acquiring a Correct Concept

A language game plays an important role in finding a correct concept. Recall that a language game (sometimes called a naming game) is used to find word agreement, so a word has no meaning if it is used in a private language and its use means nothing between agents if its meaning is not communicated. Thus, a language game is used as a tool for ostensive definitions in communications between agents.

2.2.1 Learning Process

Learning by ostensive definition contributes to general learning in language games. After setting one actor as a speaker and another as a listener, the speaker points out a thing to the listener then utters a word to refer to that thing. In this way, the listener learns a new word from the speaker.

Ostensive definition is simple to explain, but Ryan (2004) added more detail using Kripke-Wittgenstein's skeptical solution to give an account of a language game in terms of negotiations about the meaning of expression. An interesting point in this paper is negotiation by stealth occurring in the background; his epistemic analysis showed that if the listener considers the speaker to be more authoritative than him or herself, then the listener is more likely to believe the speaker. Hence, belief and authority make intuitive research an idea on how to properly manage each agent's belief in the language game. In Lindh-Knuutila et al.'s (2006) study, they counted word-meaning pairs in successful games using a counter newly defined as a belief playing an important role in mapping word-object pairs.

Some studies have defined other ways to maintain belief, such as using a specifier for keeping strengths or weaknesses (Lenaerts et al., 2005), using weights between nodes in a network (Lipowska and Lipowiski, 2012), and using the evolution of freshness in a free-scale network (Guo et al., 2014). However, all of these are similar to retaining belief based on experience from incurred successes or failures. Additionally, a utility to negotiate by stealth is used to maintain the capacity to communicate with each other as a maximizing utility considered to be a consensus of multiple listeners. Although it is not a real consensus, it is similar to the cooperative concept of listeners, thus the idea of ensuring that a game has multiple listeners in order to encourage cooperative belief is included in this study.

2.2.2 Reducing Learning Time

Meaning can gradually change to make new agreement when sharing it in a society, something which is inevitable over time. During the enactment of language games, interactions of meaning might change the meaning, so time must be considered. There are many algorithms to speed up the consensus of agreement.

The view of Lenaerts et al. (2005) on society and culture is another study that emphasizes on cultural transmission. They claimed that horizontal and oblique structures occur within one genetic generation, thus there is a difference between genetic and cultural time. Subsequently, they provided an alternative mathematical framework that incorporates those features of a cultural evolution system which are orthogonal in their model and are claimed to reduce learning time more compared to other cultural transmission systems. In computer science, an algorithm coupled with their approach creates a genetic algorithm that can incorporate those features of a cultural evolution system in a natural and clever mathematical way.

Lipowska and Lipowiski (2012) examined a naming game using an adaptive weighted network. A weight of connection for a pair of agents depends on their communication success rate and determines the roulette rule probability with which the agents communicate. In this way, they preferably select agents who have communicated successfully; this is often used with an evolutionary algorithm but not genetic evolution.

Guo et al. (2014) recommended modeling a free-scale network to improve the success rate by increasing the number of people accepting a particular opinion for a single transmission; to improve of frequency of publishing opinion, increase the number of publishers, and enhance personal ability to accept new information.

There have recently been studies investigating the use of multiple listeners (Li et al., 2013) and multiple parties (Lorkiewicz and Katarzyniak, 2014; Maity, Mukherjee, Tria and Loreto, 2013; Yuan, Chen and Chan, 2013) where one party may eavesdrop on conversations involving other parties. This approach simulates the incidence of agents simultaneously playing two roles, both as speaker and listener. However, knowledge sharing for speaking with meaning and listening with understanding in real life should also come from a social gathering (Roche, Barnes-Holmes, Barnes-Holmes and Hayes, 2002; Lipowska and Lipowiski, 2012), i.e. cultural knowledge for individuals to adjust their beliefs (likened to social introspection). Moreover, specific domain knowledge is used to promote desirable knowledge or to reduce searching for desirable knowledge. It can give the game system a better opportunity to reach desirable words more quickly. For this reason, the cultural algorithm by Reynolds (1999) is included as a social process for maintaining norms of the community and thus reducing ambiguity and variability that are also primary functions in computational linguistics (Gliozzo and Strapparava, 2009).

CHAPTER 3

FORMING CONCEPTS WITH SELF-ORGANIZING MAPS

As Lindh-Knuutila et al. (2006) suggested, language games were invented to understand how language forms in children's brains, which is also supported by Diller and Cann (2010) who stated that "… the theory demanded universal grammar to be innate so that children could universally learn language in impoverished language environments without teaching." They investigated the innateness of language from Chomsky's theories and came to believe that humans have an innate brain structure called a language acquisition device (LAD), which influences computational language today. However, other researchers (Steels, 1996; Croft, 2007) claim that language is the result of the environment and culture. Furthermore, Loreto et al. (2010) suggested that polysemy requiring the existence of two or more perceptual categories identified by the same unique word was found when utilizing the category game; it needed a two-step process to emerge and a global self-organize agreement to become stable. This situation emphasizes that cultural negotiation can improve communication. All of this has led to a combination of genetic and cultural factors in this study to construct conceptual space as a system containing each component: innate concept (newborn), social learning (interaction), and cultural learning (time dimension).

3.1 Components of the System

Belief is an important part of conceptual space. Since each new born baby (agent) does not have a prior set of beliefs, they contain empty concepts. Nevertheless, an individual will develop his/her beliefs by interacting with others in language game playing and gradually adjusts his/her beliefs about culture, which requires a certain period of time. Thus, there are three phases of language learning.

1) Newborn baby: born with an empty set of beliefs, this is the first phase during which conceptual spaces are constructed with a SOM and trained with 10 color prototypes.

2) Social learning: during this phase, beliefs are adjusted while playing language games using multiple listeners.

3) Cultural learning: the last learning phase, during which beliefs are again adjusted by exposure to cultural beliefs.

3.2 Modeling Conceptual Space

To formulate concepts, Gärdenfors (2000) proposed modeling conceptual space as a geometric structure instead of using symbolic or associationism models. With a SOM, space can be formed by a set of quality dimensions as well as, in this research, from knowledge sharing among multiple listeners and social knowledge. According to Gärdenfors, knowledge representation in cognitive science has three levels: symbolic, conceptual and sub-conceptual levels. On the symbolic level, a language is a simple kind of representation from which humans read others. When mapping the geometric structure at the conceptual level, a SOM is applied, which keeps input data as a space on nodes containing vectors of red-green-blue colors or a set of quality dimensions. Each node includes an array of concepts where each one holds a single word associated with a degree of belief. The nodes are connected as a map which represents the sub-conceptual level, the main function of which is to represent the complexity of input as general information that can be easily mapped to meaning. Thus, a SOM in this model contains knowledge from all three levels.

To create a conceptual structure, each agent needs to formulate it innately. 10 prototypes of colors (gray, blue, green, aquamarine, red, pink, yellow, white, azure, and brown) are defined to create the innate conceptual structure (the larger the number of prototypes, the more complex the knowledge will be). The innate concept is trained with the prototypes at the initial state only, before game playing. A SOM forms a conceptual space that is a repository of the physical world for the internal cognition of the agent. Conceptual space using a SOM with two-dimensional nodes is shown in Figure 3.1.

An agent maintains a degree of belief in the association between a lexeme (symbolic level) and the conceptual space. The conceptual space has three quality dimensions (denoted by red, green and blue) and holds the physical world (the colors) associated with the conceptual space.

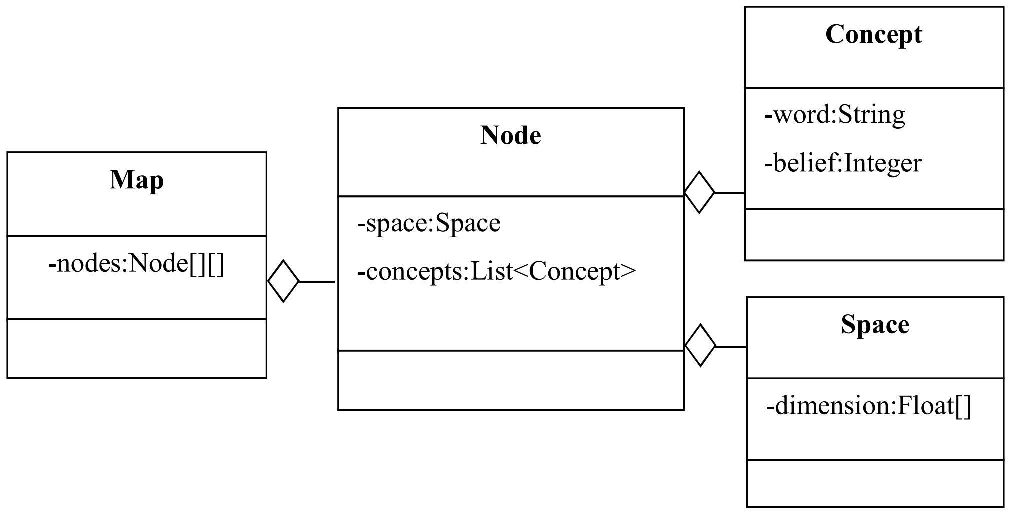


Figure 3.1 Conceptual Space Containing an SOM and Two-dimensional Nodes

3.3 Using Color as an Object

For simulations, objects are assumed to be colors with properties: red, green, and blue. The number of categories is based upon color prototypes, and in this simulation, a small number (10 object color prototypes) are used.

An object has a set of features represented by red (R), green (G), and blue (B), which can be categorized on an SOM to form conceptual space:

$$\text{Object} = \{R, G, B\}.$$

Each agent has conceptual spaces consisting of concepts constructed with an object, word (or language symbols), and its belief. A word is an utterance by a speaker and consists of discrete symbols representing limited consonants and limited vowels that when combined, become a non-limited word. When each word is bound with a belief, it is counted as a success when playing the game and may be referred to as a so-called confidential experience for each agent. A belief has a limited number of settings ranging from 0 to 20 and if this number is high, it is a high belief, else if the number is low, it is a low belief:

Concept = {Object, Word, Belief}.

A given set of topics randomized for agents playing a language game is shown for both the speaker and listeners:

Topic = {(Object$_1$,), (Object$_2$,), …, (Object$_n$)}.

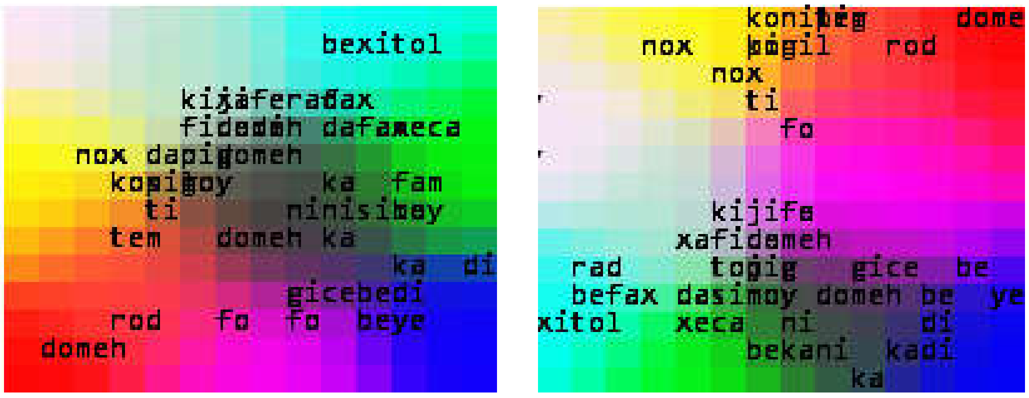


Figure 3.2 Two Examples of Conceptual Spaces Mapped to Object Meaning

3.4 The Language Game Algorithm

Playing a naming game is a process of finding a proper name to identify an object. A speaker and listeners are random selected, after which the speaker is the first one to utter a word that is the best belief for the object topic, and the listeners either accept the speaker's word for the object topic or reject it. This process involves searching for a word in conceptual space.

The algorithm is divided into three main phases of language learning: newborn, social (self), and cultural, but the phases can occur simultaneously. A language game can be described as follows:

(Newborn)

The agents' conceptual map is trained with three–dimensional color data vectors using a SOM to classify the color categories.

(Social Learning)

1) The speaker is chosen randomly from a group of agents. The agent is arbitrarily assigned the role of speaker, while a number of agents are also randomly chosen as listeners (e.g. 60% of the remaining agents).

2) The topics are initially generated and randomly chosen in the game.

3) The speaker searches for the belief concept (a node in its SOM) that best matches the topic. This is called the best-matching unit (BMU).

4) If nothing is found, the speaker looks at the cultural memory and after searching, selects the best match in the cultural memory. If it cannot be found, the speaker searches in the neighboring nodes; this situation can produce synonymy, which is a natural process in human language. If there is still no word found, the speaker will generate a new word and keep this word mapped to the best node. Finally, the speaker utters this word to the listeners.

5) The listeners search this topic in their memory in the similar fashion to the speaker. However, they integrate the three levels of belief: self, social, and cultural.

6) When the listeners receive a word from the speaker, they search their self-memory. If the number of the listeners who find the word is more than a pre-specified number (e.g. 50% of listeners), the uttered word of the speaker is accepted and this game is considered to be successful. The unknown listeners update the beliefs associated with that word in their own BMUs.

7) If the sixth step fails, searches are continually made in neighboring nodes (in the listeners' SOMs). By doing this, a synonym might also be created, as in natural language. If the number of listeners finding that word is more than the pre-specified number, this game is considered a successful game, and the unknown listeners update their beliefs.

8) If the seventh step fails, searches are continually made across all nodes. If the number of listeners who find that word reaches the pre-specified number of listeners, the unknown listeners update the knowledge following the known listeners.

This game is considered a successful game; otherwise the game fails. For a successful game, both speaker and listeners increase their belief counters by one, while for a failed game only the speaker decreases its belief counter by one. The maximum belief is set at 20 and minimum belief is set at zero.

(Cultural Learning)

Meanwhile, a cultural algorithm is applied to adjust cultural belief.

3.5 Searching for Words in SOMs and the Cultural Repository

After an SOM has been trained 1,000 times with random prototypes, a random topic, i.e. a color vector, is shown to the speaker and the listeners. The process of searching for the best word is performed at four levels: searching in the best node, the cultural repository, neighboring nodes, and then all nodes. The speaker searches in the first source (the BMU). This node may contain many concepts in which each concept stores a word associated with its belief. Only the best belief is selected. Recall that the best belief is defined by a count each time a game succeeds, after which it is incremented by one.

If there is no word in the best node, the speaker searches in the cultural repository, which is maintained as a hash collection. If it finds the mapping node in this collection, the speaker keeps this word and associates it with the node. However, the word may not be found in the cultural repository, then the speaker will search for a word in its neighboring nodes within radius *R*. For effective communication, the radius is set at one or two neighborhoods. If the speaker cannot find a word in the neighboring nodes, it will search all nodes in the map. If it still cannot find any word, a new word is generated, mapped to its BMU, and finally uttered to the listeners.

The topic that has been shown to the speaker is also shown to the listeners. A listener performs searching in a similar way to the speaker, except that it does not look at the cultural memory. Thus, listener searching has only three stages: the BMU, neighboring nodes, and then all nodes. Moreover, at each stage, the number of agents who find the word is pre-specified. If the number of those listeners is greater than the value, it is considered to be a successful game, and the word is spread to all listeners who did not find it. If the listeners cannot find a word in any state, the communication fails. However, they must add the speaker's word to their BMUs, and the speaker decreases its belief counter by one for this word. In the case of a successful game, the speaker and listeners increase their belief counters for that word by one.

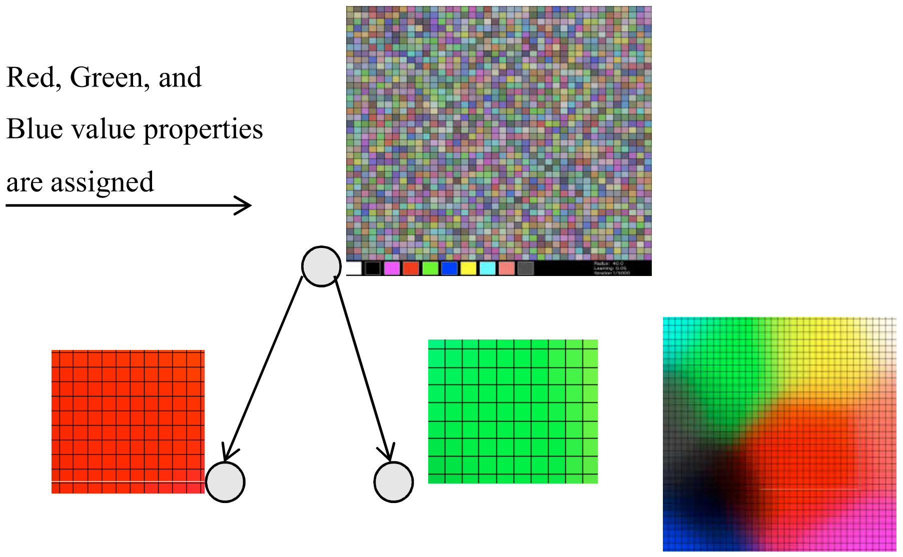


Figure 3.3 Color Properties Assigned to Different Nodes in a SOM

Figure 3.3 shows a visualization and classification of tree-structure data. The root of the tree represents a physical world in which random colors is assigned to the player. The lower tree shows the color prototypes used for conceptual space (in this example, there are ten color prototypes, i.e. red, green, blue, pink, yellow, etc.).

3.6 Lexicon in Nodes

The word generation process uses all of the English characters (A – Z). A word has a length of two to six characters, interleaving consonants and vowels, e.g. 'CABAKI'. Word generation is performed by the speaker, and Figure 3.4 shows words generated by a speaker on a SOM map consisting of 14 x 14 nodes to reduce computational resources while still sufficient to generate suitable words in simulated communications.

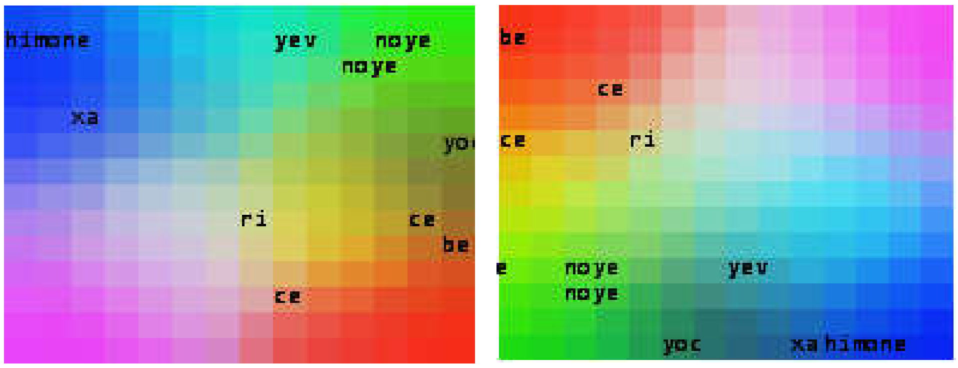


Figure 3.4 Possible Words on a SOM

3.7 Improve Ambiguity with Flexible Search in Conceptual Space

Even fixed searching of conceptual space can help make agents come to an agreement and can be used for communication, but polysemy can occur easily with closeness in meaning. To remove this ambiguity, flexible searching in conceptual space was incorporated. As Lindh-Knuutila et al. (2006) showed, the search radius (R) in a neighborhood has an effect on word ambiguity in that a higher radius range ($R > 4$) results in more ambiguity, but less range achieves more effective meaning, and an R value between 1 and 2 has more effect on the number of played games. These results point towards how to set a flexible range for searching in the neighborhood for agents to obtain effective meaning when playing language games.

The new algorithm for the flexible search was applied to both the speaker and the listeners, and neighboring node searching used a flexible radius, i.e. the initial radius value was set at 0.2, and increased by 0.2 every round up to a limit of 1.8. Searching stopped if the best word was found. The speaker then uttered the selected word to the listeners. Furthermore, the search algorithm for the listeners was similar to that of the speaker.

3.8 Evaluation Measures

To evaluate agent learning, four measures were utilized: communication success, coherence, specificity, and lexicon size. The first two measures were taken from DeJong (2000) and Lindh-Knuutila et al. (2006), respectively.

3.8.1 Coherence

The basic hypothesis of lexical coherence is that a great percentage of the concepts expressed in the same text belong to the same domain. One concept can occasionally be conveyed by many words, i.e. synonymy, as shown in Figure 3.5. This is natural in human languages.

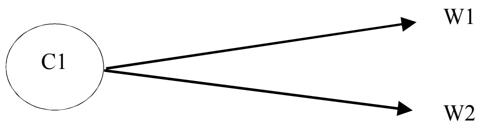


Figure 3.5 Measuring Coherence

Coherence is a measure of whether a certain word is utilized coherently among the agents to denote a certain meaning in the community. It is calculated from the rate of average word reference per average no word reference to the topic of the agent. Coherence allows the disambiguation of ambiguous words by associating domain-specific sense to them.

Table 3.1 shows concepts in each agent and their coherence. The black circles indicate words agreed with other agents. Concept 1 has two different words (*W1* and *W2*). To calculate the coherence of each agent (coh(A_i)), the number of agreed words are summed and divided by the total number of concepts, as shown in the following formula:

$$coh(A_i)=\frac{W_c}{\sum_{i=0}^{n} A},$$

where w_c is the number of agreed words (black circle), and A is the total number of concepts in the games.

Table 3.1 Coherence Calculation

Agent	Concept 1	Concept 2	Concept 3	Concept 4	Coherence
Agent1	W1 ●	W3 ●	W4 ●	W6 ●	1
Agent2	W1 ●	W3 ●	W5	W7	0.5
Agent3	W2	W3 ●	W4 ●	W6 ●	0.75
Agent4	W1 ●	W3 ●	W4 ●	W7	0.75
Frequency	3	4	3	2	
Coherence	0.75	1	0.75	0.5	

In some cases, one word may convey different meanings or concepts, which may cause ambiguity. Figure 3.8 shows that concepts 1 and 2 are associated with word 1; a polysemy occurs here whereas it does not occur for concept 3 referring to word 2 (W2) or concept 4 (C4) referring to word 3 (W3).

3.8.2 Specificity

As De Jong (2000) said, “Specificity indicates to what degree the words an agent uses determine the referent that is the subject of communication.” Specificity decreases if two meanings refer to the same word, thus it is the degree of polysemy in the lexicon: the higher the specificity, the less polysemy there is.

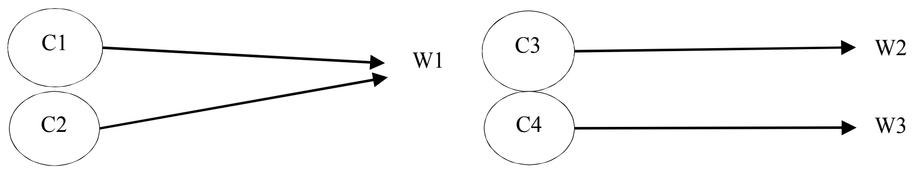


Figure 3.6 Measuring Specificity

The specificity of the population spec(A$_i$) is then defined as the average specificity of the agents:

$$\text{spec}(A_i) = \frac{n_s^2 - \sum_{k=1}^{n_s} f_k}{n_s^2 - n_s},$$

where n_s represents the number of concepts and f_k is the frequency of the word related to those particular concepts. This formula specifies the degree of polysemy.

Table 3.2 shows various specificities under different circumstances. When agent 1 has two concepts (1 and 2) referring to the same word (W1), while concepts 3 and 4 have a single reference since they refer to different words, W2 and W3, respectively.

Table 3.2 Concepts of Agents and Their Specificities

Agent	Concept 1	Concept 2	Concept 3	Concept 4	∑f	Specificity
Agent1	2	2	1	1	6	0.833
Agent2	1	3	3	3	10	0.5
Agent3	1	1	1	1	4	1

To select a word for a concept while an agent has many words associated with it, the selected word is the one with the highest values of coherence and specificity. Word ambiguity has been considered to be coherence in some papers (Lorkiewicz and Katarzyniak, 2014; Lorkiewicz, Kowalczyk, Katarzyniak, and Vo, 2011).

3.8.3 Word Count

At the end of a game, the average size of the lexicons was calculated. In counting the number of words in lexicons, words having zero belief (i.e. were not used in a successful communication) were also included. The measure can be calculated as follows:

$$word_{count} = \sum_{i=0}^{n} node\left(\sum_{j=0}^{m} concept_j\right)_i,$$

where *m* is the number of concepts in a node, and *n* is the number of nodes on the map. A smaller word count indicates that the communication is more effective than a larger size.

3.8.4 Success Rate

Success rate was computed after every 10 games and calculated as a sliding window average as follows:

$$succ_{rate} = \frac{1}{m} \sum_{i=0}^{m} \left(\frac{1}{n} \sum_{j=t-n+1}^{t} x_j \right)_i ,$$

where x_j is the successful communications evaluated after every 10 games, *n* is the number of sliding windows, *m* is the total number of agents, and *t* is the number of games. The success rate indicates the effectiveness of the communications.

CHAPTER 4

CULTURAL ALGORITHMS AND THEIR USES IN LANGUAGE GAMES

Evolutionary computation (EC) is a mechanism extracted from the understanding of how natural systems evolve to solve complex computational problems. EC focuses on the process of natural selection and genetics in populations. A Cultural Algorithm (CA) works in the same way as the EC process, but focuses on knowledge selection in the population that is transmitted from generation to generation. In this chapter, a general description of a cultural algorithm is provided. Spaces in the cultural algorithm can be maintained by language games similar to other evolutionary processes.

4.1 The Basics of the Cultural Algorithm

Reynolds (1994) is the pioneer of CAs. He proposed the method as an extension of the version space algorithm (a binary string-based genetic algorithm). The version space is the generalization of individual solutions communicated as cultural knowledge in the form of schema patterns (string of 1's, 0's, and #'s, where '#' represents a wildcard). His idea was inspired by Renfrew's thinking (Renfrew, 1994) by what the latter refers to as a world map: the internalized knowledge or symbolization of an individual's past experiences and forecasts concerning future experiences. The map can be merged, generalized, and specialized to form a group of maps via evolution that in human societies is called cultural transmission.

To model the algorithm, Reynolds extracted knowledge from an individual, which he defined as a set of traits, and generalized their experiences later to become the map suggested by Renfrew. Traits can be represented by symbols that can be modified, exchanged, added, or possibly lost between individuals by means of a variety of socially motivated operations. The map can also be modified over time based upon experience, thus traits and map can evolve as the result of group experiences.

The process of evolution is like natural selection defined as rules for evaluation. The set of evaluated traits is most of the general beliefs in a group of populations called the belief domain, which later Reynolds named the population space (Reynolds, 1999). The population space is promoted via acceptable functions to the upper level and then can be merged with currently existing group maps in the belief space if the conditions for one or merging operations are met via adjusted functions. Hence, the belief space is specific knowledge separately stored in the cultural knowledge of individuals or a population. This belief space is used as guidelines or norms that everyone agrees with for individual action, and these guidelines are defined as an influence function that has influence or feedback to control the individual. While belief space has influence on a population, the population variates itself before the selection start again. After each step in time, the belief space will become more specific. The process of CA as described is illustrated in Figure 4.1.

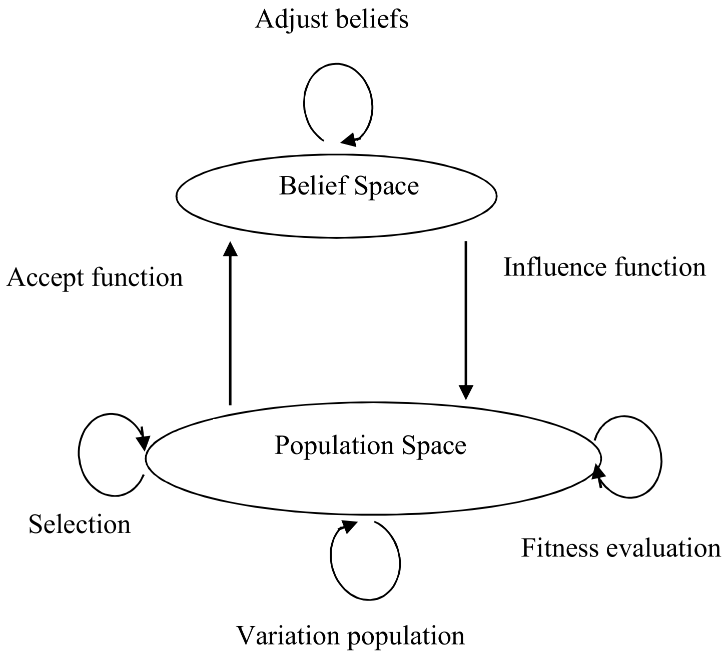


Figure 4.1 The General Framework of a Cultural Algorithm

The pseudocode for the above general model is given in Figure 4.2.

```
Begin
      t = 0;
      Initialize Population P
      Initialize Belief Space B
      while (condition is true)
            Evaluate($P_t$);
            Adjust($B^t$, Accept($P^t$));
            Variate($P^t$, Influence ($B^t$));
            t = t + 1;
            Select $P_t$ from $P_{t-1}$
      end
End
```

Figure 4.2 General Cultural Algorithm Pseudocode

4.2 The Space-Guided Evolutionary Algorithm

Various CAs can be implemented depending upon how the population space and the belief space are represented, such as evolutionary algorithms (EAs) as well as network, logic, and set theory (Reynold, 1994), but the most common is EAs due to their ability to evolve using natural selection, as can be seen in optimization problem applications (Brownlee, 2011). One reason for this choice is that EA can mimic natural evolutionary processes such as selection, variation, and reproduction to produce a new generation in an accelerated version of natural evolution. For CAs, spaces contain knowledge, experience, or belief of an individual that has evolved via transmission from generation to generation.

Genetic algorithms (GAs) are a kind of EA that use chromosomes to represent population traits. This representation has become one of the important features of CAs that can be used as traits to represent belief corresponding with population experiences. A GA's chromosomes can be modified (known as mutation) or exchanged (known as crossover) between individuals based upon population experience over the evolutionary timeframe. The process of GA's selection is a technique to achieve a new desirable generation. One method is to use roulette wheel selection, and a CA can use this technique to produce a new generation with new belief coming from the population space. Subsequently, knowledge from information acquired by generations is stored at the higher level of belief space, which is accessible by the current generation. Interaction between the two spaces gradually progresses in an evolution-like process.

4.3 The Space-Guided Language Game

To apply CA to a language game, the agents are the population with the primary source of knowledge (or normative knowledge) retained in the population space, while secondary knowledge in the belief space holds the cultural belief.

Knowledge is represented as belief in words with different degrees occurring during communication among the social agents. When a speaker utters a word to listener, the belief of the word will change, and then the population space is also changed. With GAs, the trait is in the chromosomes but the language game keeps the trait in the belief. In the same way that a GA can solve an optimization problem, a language game can achieve this in a similar way, by which they are often used to solve meaning problems.

The internal structure of conceptual spaces retains knowledge in nodes implemented in a SOM. Each node contains the space mapping of a physical word (red, green, and/or blue) and a list of concepts. The concept is composed of the pairing of a word with the degree of belief that change while play the language game is being played, thus the two spaces are guided by the language game.

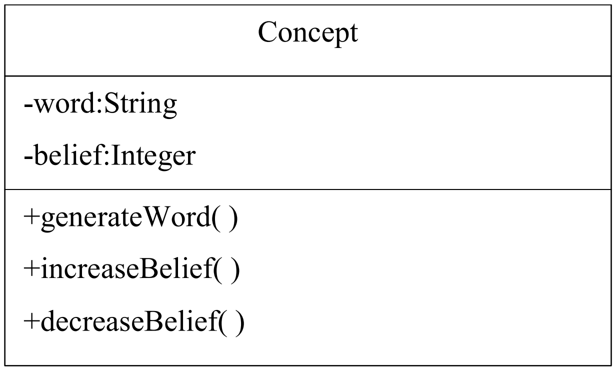


Figure 4.3 The Concept to Maintain the Belief of a Word

Figure 4.3 shows the concept that is used to maintain the belief of a word mapped to a space. The belief can be changed by an agent's experience and success rate when playing the game.

4.4 Using a CA in a Language Game

To incorporate a CA in a language game, a protocol is set to allow the two spaces to interact and to exchange information. Hence, CAs in language games can be defined as

$$CA = \{P, V, B, f, \text{Accept}, \text{Adjust}, \text{Influence}\},$$

where *P* is the population or players, *V* is a variate function, *B* is the belief space, *f* is the performance function representing the problem-solving experience of an individual, *Accept* is the acceptance function, *Adjust* is the adjust function that adjusts or updates the belief space, and *Influence* is the influence function that is used to influence the variation function. However, the players do not reproduce in order to maintain the same population in games, so the selection is not applied. The pseudocode for the proposed CA for language games is shown in Figure 4.4.

```
Begin
    $Game^t$=0;
    Initialize $P^t$
    Initialize $B^t$
    while (condition is true)
        Evaluate(P($COH^t$, $SPE^t$));
        Adjust($B^t$, Accept($P^t$));
        Variate($P^t$, Influence ($B^t$));
        $Game^t$ = $Game^t$ + 1;
    end
End
```

Figure 4.4 The Proposed Cultural Algorithm for Language Games

Language games start at game 0 (*$Game^t$*). The agents are generated (*P^t*) with empty belief, and the belief space (*B^t*) is also empty. When a game begins, the agents' knowledge is evaluated with a performance function that computes the two fitness measures: coherence (*COH^t*) and specificity (*SPE^t*). The results of these evaluations are determined by the acceptance function (*Accept(P^t)*). The agents' knowledge with the best fitness is considered to be acceptable and is promoted to the belief space with the adjust function (*Adjust(B^t, Accept(P^t)*). The adjust function uses the acceptable knowledge to update the belief space which is shared by the population. The new belief space has an influence on listeners' beliefs via the influence function (*Influence(B^t)*) when the speaker chooses this knowledge. As described earlier, the speaker selects a word from the cultural belief only if it is not found in its BMU and neighboring nodes, thus the selected word from the belief space that maps to the topic has influence on the population. A listener will accept the knowledge or adjust its belief with the variate function, i.e. a guided variation (Mesoudi, 2011). After each step of the game, the two sources of knowledge are updated, and the belief space and the individuals' normative knowledge are recalculated. Consequently, the language games promote the specific knowledge in the belief space and broker agreements among the agents.

CHAPTER 5

EXPERIMENTAL RESULTS

In the game evaluations, four measurements were made: the comparative performance between single and multiple listeners, the effect of the number of agents, the effect of the number of listeners, and the effectiveness of including social knowledge. Moreover, flexible searching was incorporated in the search algorithm whereby testing first searched either the BMU or used the CA, whichever one was more effective.

5.1 Single Listener vs Multiple Listeners

The first comparison of using a single listener and multiple listeners was tested using four measurements: coherence, specificity, game success rate, and word count. Figure 5.1 shows a comparison of coherence and specificity between a single listener and multiple listeners. The results are averaged across 10 simulations using 50 agents, 60% of which were randomly selected as listeners. We can see that the graphs of multiple listeners rise sharply at the beginning and become stable as the games developed, which indicates that using multiple listeners was much better than a single listener since knowledge was spread and exchanged among the agents.

The results on success rate and word count are shown in Figure 5.2. It is evident that the success rate of using multiple listeners (left graph) increased sharply at the beginning, while that of using a single listener increased gradually and reached only around 25%. In the right graph showing word count, using multiple listeners created a much smaller word count, indicating the ability of using words to communicate successfully using a smaller set of words. In addition, a small word count reduced word ambiguity and variability. It is clear that using multiple listeners was more effective than using a single listener. In further experiments, only multiple listeners were examined.

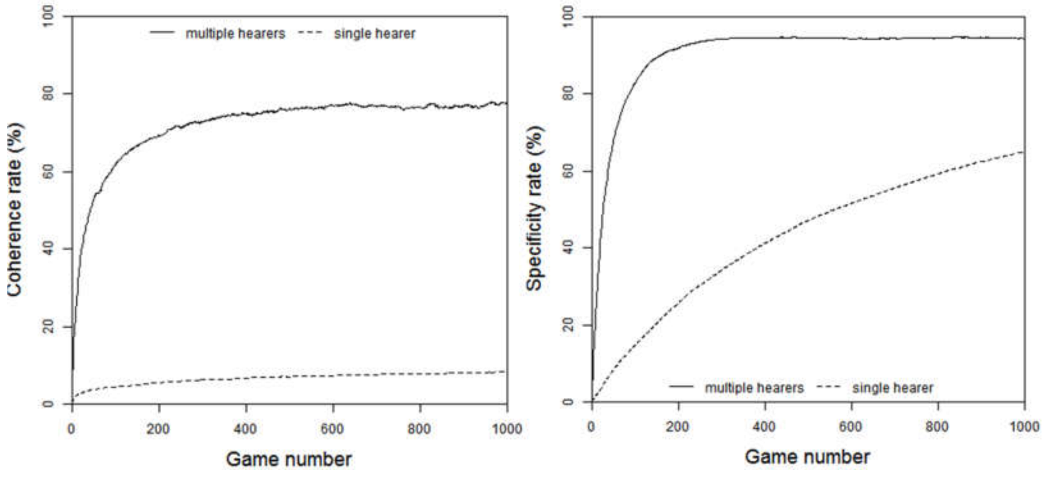


Figure 5.1 Coherence (left) and Specificity (right) with 50 Agents

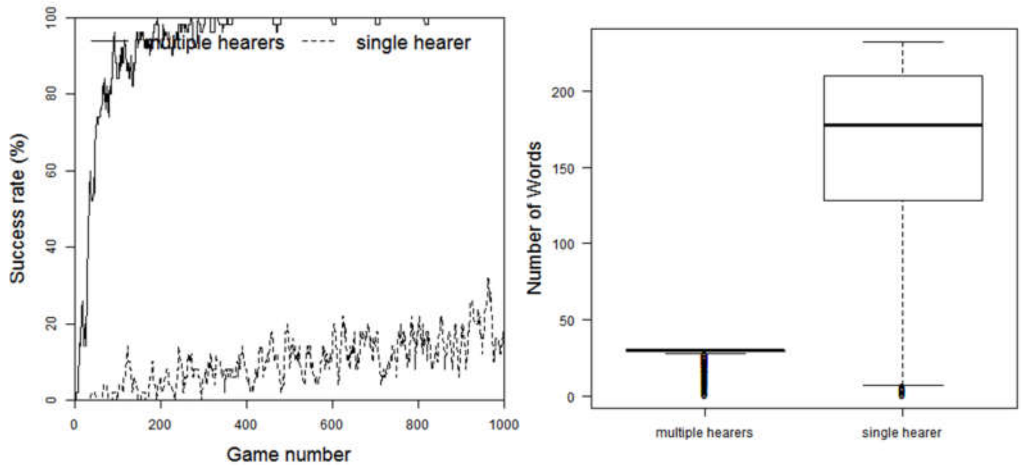


Figure 5.2 Success Rate (left) and Word Count (right) with 50 Agents

5.2 Effect of the Number of Agents in Games

When more agents are involved in the games, the ambiguity and variability were expected to increase. Figure 5.3 shows coherence and specificity rates of the different number of agents as the games proceeded. In general, as shown in the previous section, coherence and specificity rose steeply early on and slowly converged to a certain level as the games progressed. Three sizes of agent community: 5, 10, and 20 (60% of those were selected as listeners) were selected to play the language games.

We can see that a small group of agents communicated much more efficiently than a larger group since the smaller group created less variability in the words generated and found it easier to communicate. To improve on word variability and ambiguity, a better method was needed which is reported on in next section.

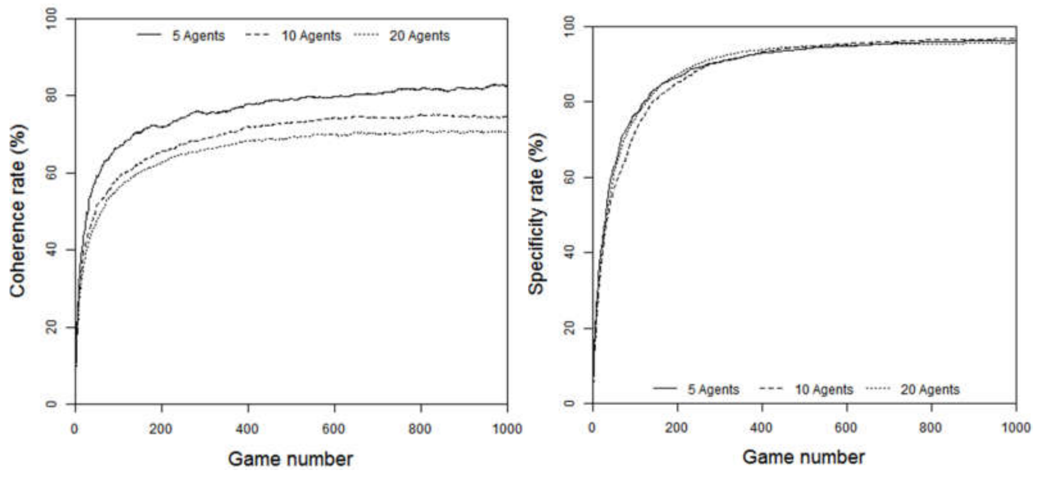


Figure 5.3 Effect of the Number of Agents

5.3 Effect of the Number of Listeners

In this section, the study of the effect of the number of listeners by varying the percentage of agents selected as listeners from the total of 50 agents was investigated. The results in Figure 5.4 show that more listeners had better coherence and specificity and at the same time climbed to a steady state more quickly than fewer listeners because more listeners generated more knowledge sharing. This means that the interpretation of object meaning depended more on interpretations by others than the agent itself.

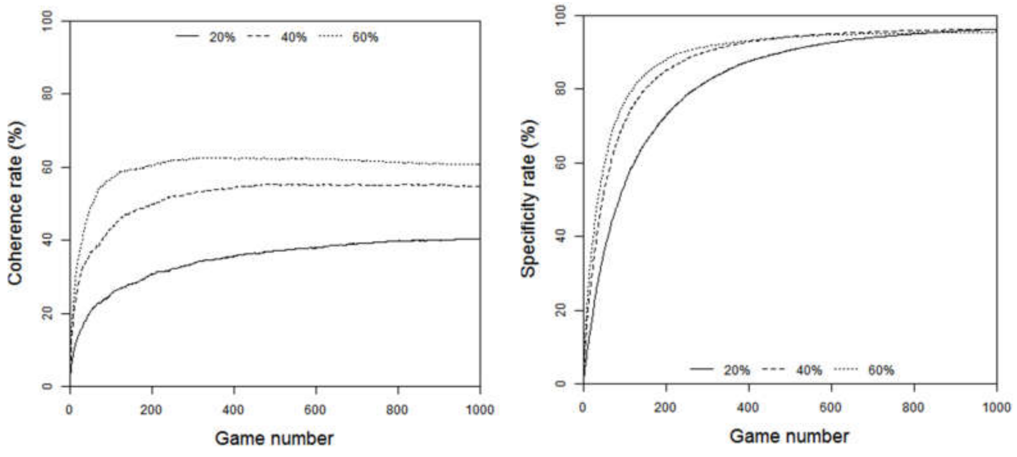


Figure 5.4 Effect of the Number of Listeners

5.4 Effectiveness of Social Knowledge

So far, the examination of language games without the inclusion of social knowledge was implemented by the CA. In this experiment, social knowledge as described in the proposed method was included and its effectiveness studied. We can see from the results in Figure 11 that using CA yielded more coherence and slightly more specificity, which means that social knowledge allowed agents to reach agreement more efficiently as expected.

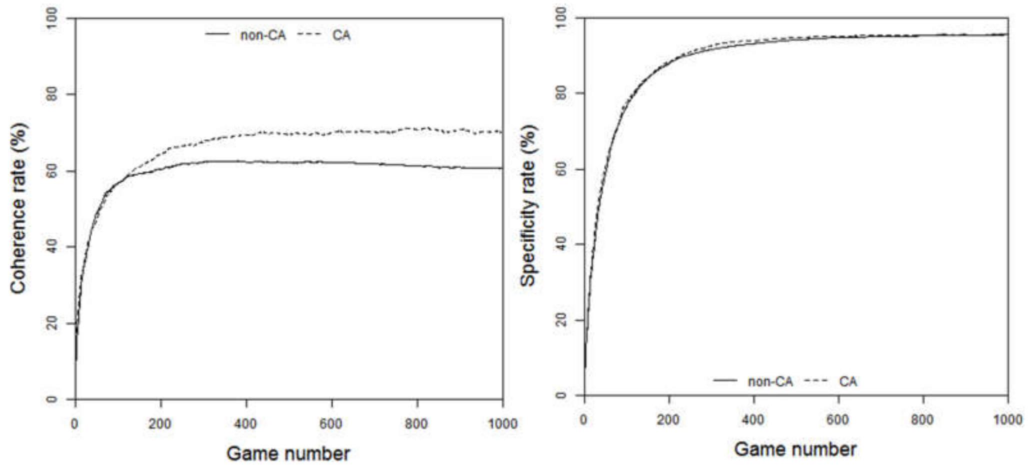


Figure 5.5 Effectiveness of Social Knowledge on Coherence and Specificity

The results on word count, shown in Figure 5.6, show that the games using the CA generated lower word counts than those without it. The results indicate that agents in a society holding strong beliefs extracted them from it and were more confident in communication if the speaker chose a standard belief about a word-object pairing from the cultural memory when it encountered an obscure situation.

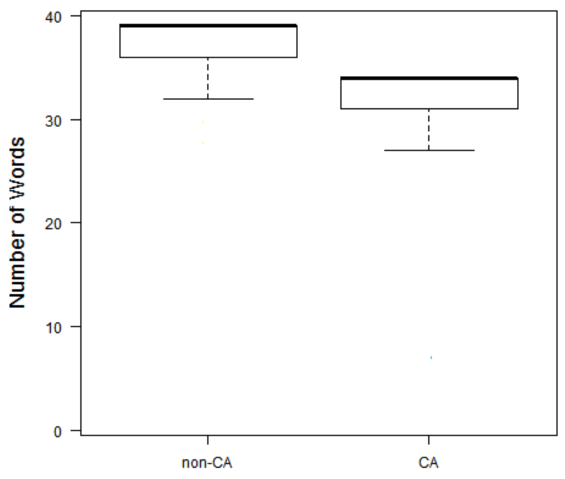


Figure 5.6 Effectiveness of Social Knowledge on Word Count

5.5 Effectiveness of Flexible Search on Conceptual Space

In this section, the effectiveness of the flexible search on conceptual spaces is reported, with the expectation that it gives more effective communication among social agents.

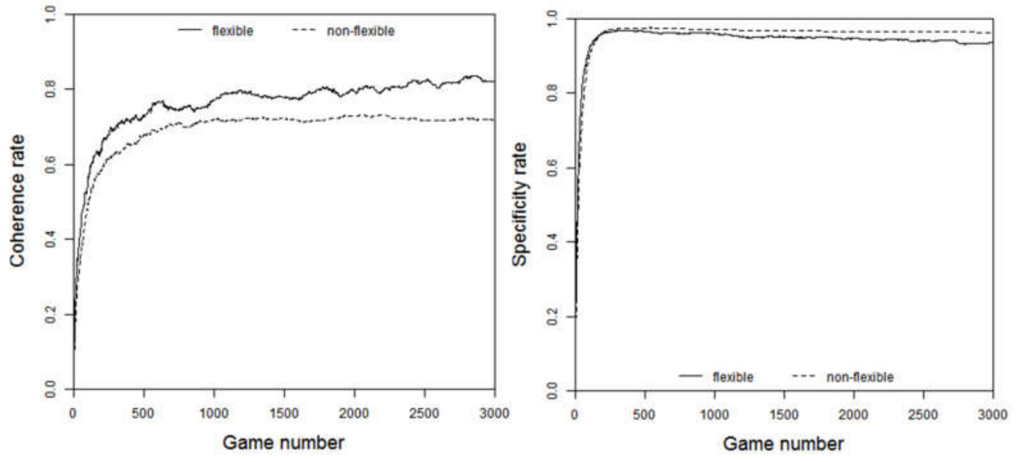


Figure 5.7 Coherence and Specificity Using the Flexible Search

In Figure 5.7, at game 100, the coherence graph shows that using the flexible search space rose to a higher rate than not using it. After 3,000 games, there was only a 20% improvement caused by the agents having more choice to interpret a word's meaning. In terms of specificity, using a fixed search space produced only a slightly higher rate after game 250 through to the end of the game playing because the increase in meaning choices probably caused a word to be used to describe multiple objects on different nodes.

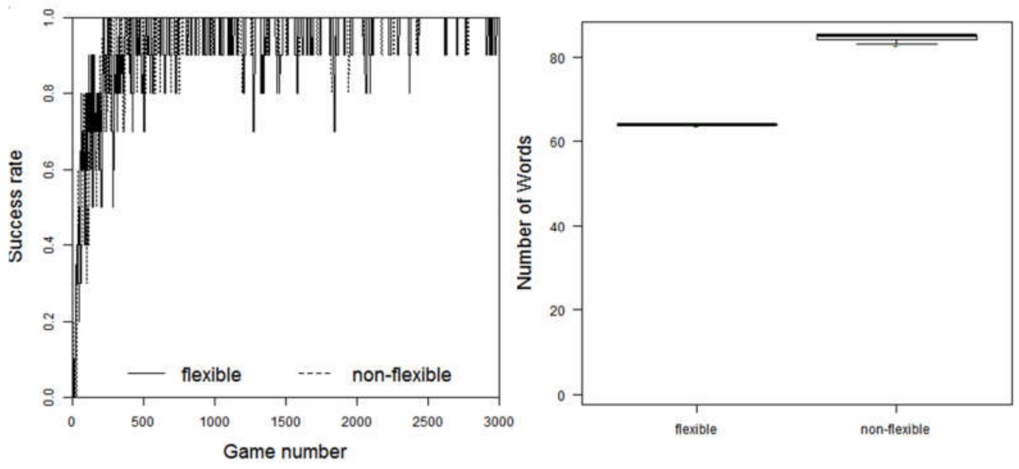


Figure 5.8 Success Rate and Word Count Using the Flexible Search

On the success rate graph (Figure 5.8), it is evident that whether using the flexible concept or not is not much different. Both graphs show a good success rate from the beginning of the game onwards. For the word count, using the flexible concept reduced the number of words.

5.6 Effectiveness of the Flexible Search with Cultural Belief

The best concept voted for by agents will be promoted to the belief space or norm for that belief for the society. As can be seen in the left-hand-side graph in Figure 5.9, the coherence graph using the CA has a higher rating and smoother line, but after a large number of games had been played, the two lines tend to converge. This is the effect of cultural belief supporting the agents with norm concepts since the beginning of the games. In the long run, all agents learned all or most of the topics, thus the graphs converge. For the specificity rate in the right-hand-side graph in Figure 5.9, the results show that the line with the CA remains steady after game 200, while the method not using cultural knowledge levels off.

As expected, in the success and word count graphs shown in Figure 5.10, using cultural knowledge obtained a constant success rate of 1 from game 250 onwards and also yielded a smaller word count. Therefore, including cultural knowledge in addition to the use of the flexible concept in the games improved the performance further.

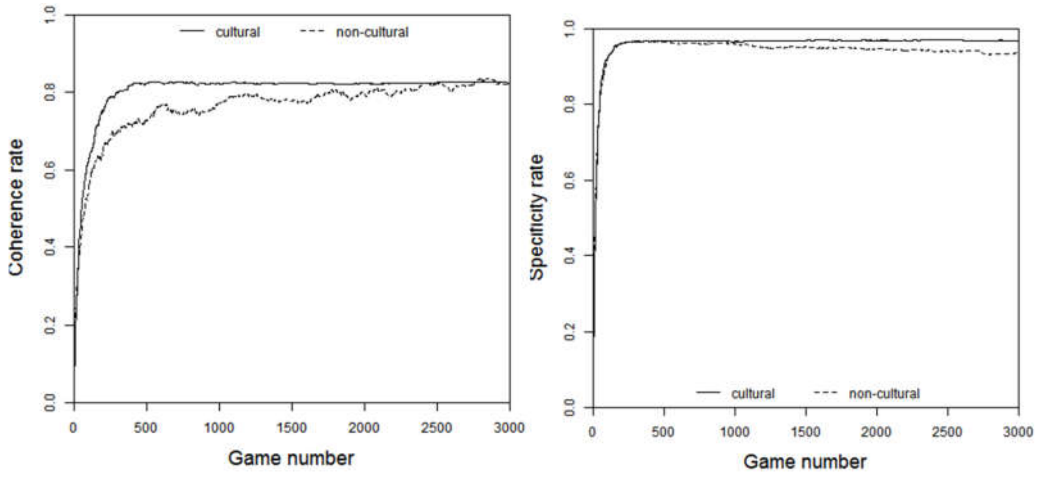


Figure 5.9 Coherence and Specificity Using the Flexible Search with Cultural Knowledge

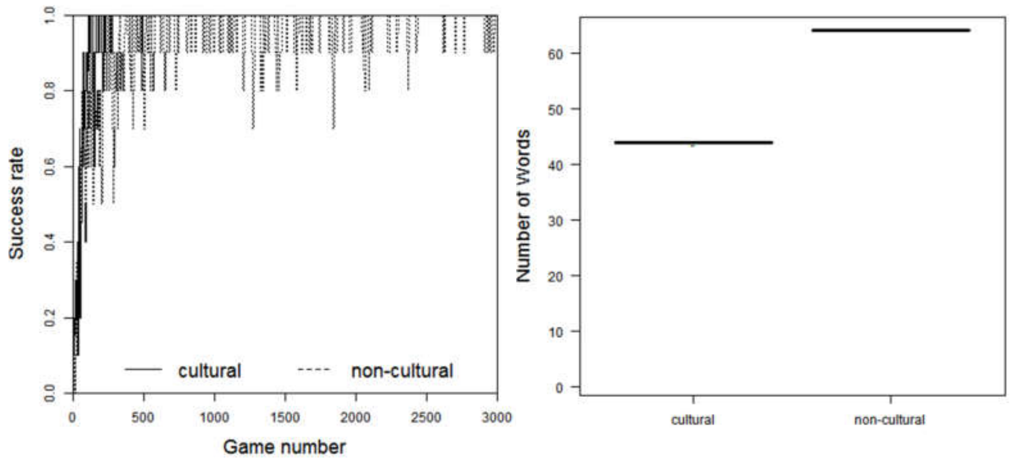


Figure 5.10 Success Rate and Word Count Using Cultural Knowledge

5.7 Effectiveness of Applying Search with the CA Prior to the BMU

The previous simulations were tested on searching the BMU before searching with the CA. Thus, the question may arise, “Is it better if an agent searches spaces with the CA prior to the BMU?” from the presupposition that the CA is the most confident belief of the society of agents, while the BMU is an individual’s belief which may lack agreement with others because it is not based on a consensus.

The presupposition was correct for all cases, as can be seen in the following figures. These simulations showed that when the agents search using the CA first, selecting the correct word was more likely than searching in the BMU first.

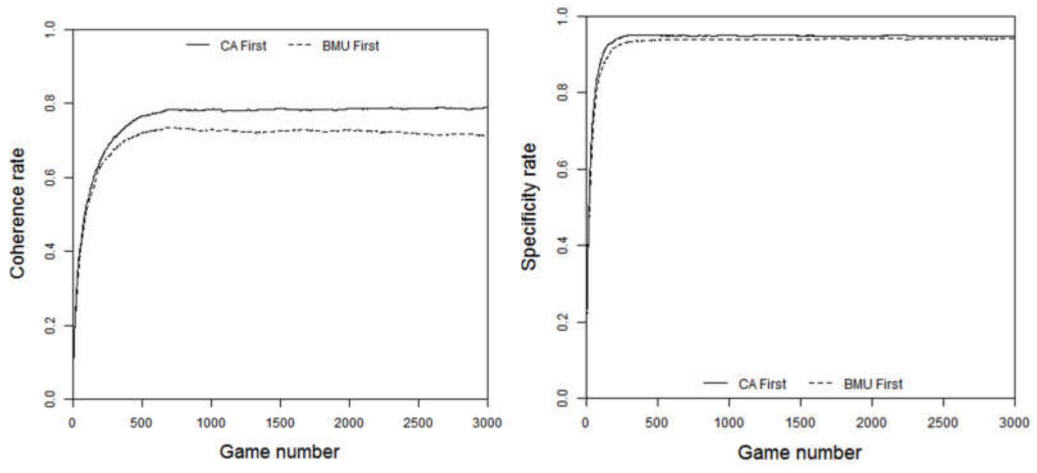


Figure 5.11 Coherence and Specificity when Searching with the CA and in the BMU

Figure 5.11 shows that coherence was higher when agents searched with the CA first compared to when agents searched in the BMU first, which was also the case for specificity, albeit only at a slightly higher rate.

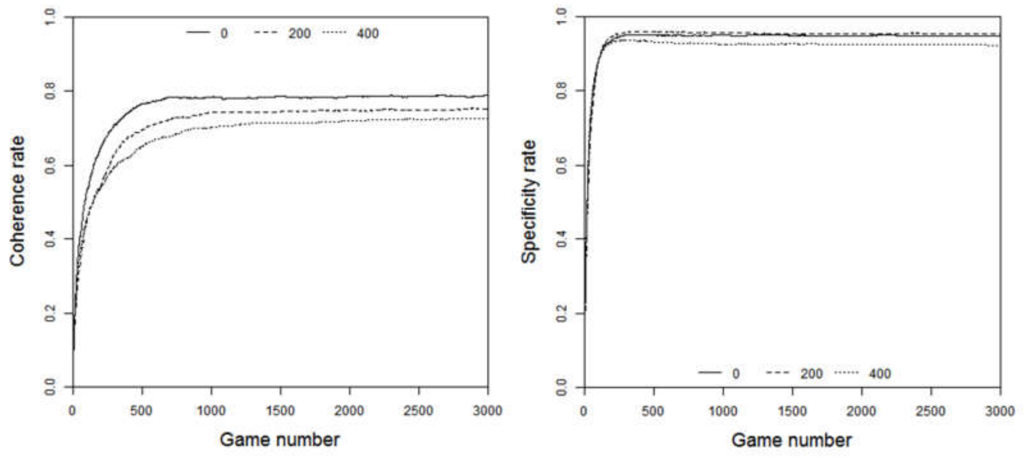


Figure 5.12 Coherence and Specificity when Searching with the CA First at Various Starting Points

Since using the CA first to search was better, it is of interest to discover whether searching should start earlier or later while playing the games, which is influenced by the presupposition that at the beginning of game playing, the belief space in cultural memory is empty or has little belief. For testing this assumption, the simulation shown in Figure 5.12 indicates that there was an effect in that searching using the CA first at the beginning of game playing had a higher effect than using CA first at a later time. The significance is obviously seen in the coherence graph.

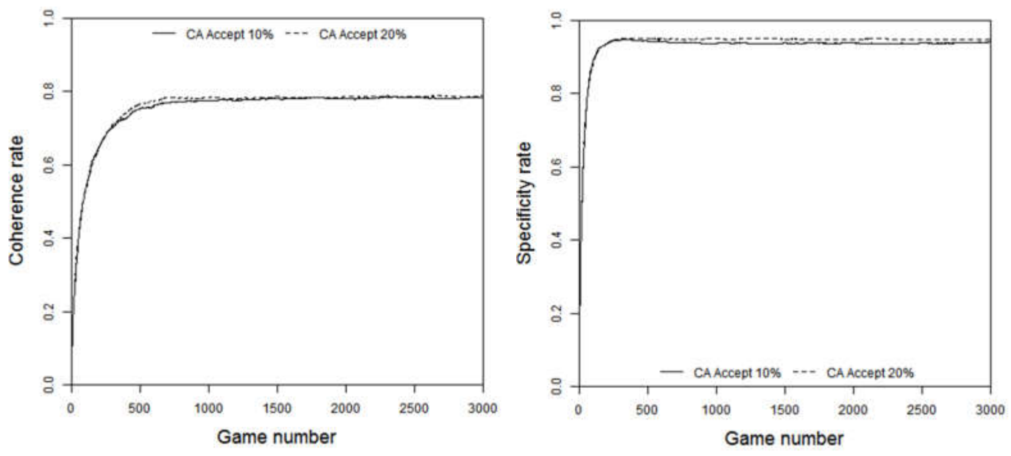


Figure 5.13 Coherence and Specificity when Searching with the CA First with Different CA Acceptance Criteria

Lastly, different CA acceptance functions were tested. The test compared the different percentages of the acceptance function used in the CA: 10% and 20% by their effect on coherence and specificity, and the results shown in Figure 5.13 demonstrate that was no significant difference.

So far, it can be concluded that using the CA to search first before searching in the BMU was more efficient because the belief space in the CA is the belief that is extracted from the social belief even if searching at beginning of game.

CHAPTER 6

CONCLUSIONS

Language games attempt to find meaning for objects by using mutual agreement, and autonomous agreement among agents is important for improving communication. In this research, language games were proposed to share agreement in populations using the three levels of belief: conceptual space implemented with a SOM to form individual belief, multiple listeners to create social belief, and a cultural algorithm to normalize cultural standard belief.

The evaluation results show that using multiple listeners yielded fast convergence to mutual agreement in a society of agents, a low lexicon size, and a higher success rate, especially with a large number of agents. In addition, using the cultural algorithm helped the speaker select words from the cultural repository when the speaker needed help and had a positive impact on the word-belief of the listeners. Moreover, an improved flexible search in conceptual space was able to make finding agreement among the agents more effective.

In future research, sharing belief, either social or cultural, should be investigated in more detail to help process autonomous learning more quickly and correctly. Moreover, learning by agents in language games could be viewed as an evolutionary process for finding a better solution so that there is no end to the process when finding meaning for an ambiguous word by extending the process to find the optimal solution, much like the general application of evolutionary algorithms in computing.

BIBLIOGRAPHY

Aitchison, J. 2012. **Words in the Mind: An Introduction to the Metal Lexicon.** 4th ed. Hoboken, N.J.: John Wiley & Sons.

Beule, J. D. and Bleys, J. 2010. Self-Organization and Emergence in Language: A Case Study for Color. In **The Evolution of Language: Proceedings of the 8th International Conference** Utrecht, Natherlands: World Scientific. Pp. 83-90.

Brownlee, J. 2011. **Clever Algorithms: Nature-inspired Programming Recipes.** Retrieved from https://users.info.uvt.ro/~dzaharie/ma2016/books/ CleverAlgorithms_Brownlee_2011.pdf

Chella, A.; Frixione, M. and Gaglio, S. 2001. Conceptual Spaces for Computer Vision Representations. **Artificial Intelligence Review.** 16: 137-152.

Croft, W. 2007. Language Structure in its Human Context: New Directions for the Language Sciences in the Twenty-First Century. In **Proceedings of the 7th International Conference on the Evolution of Language.** Utrecht: World Scientific. Pp. 75-82.

DeJong, E. D. 2000. **Automation Formation of Concepts and Communication**. Doctoral dissertation, Vrije Universiteit, Brussels, Belgium.

Diller, K. C. and Cann, R. L. 2010. The Innateness of Language: A View from Genetics. In **Proceedings of the 8th International Conference on the Evolution of Language.** A. D. Smith, M. Schouwstra, B.D. Boer, & K. Smith, eds. Utrecht: World Scientific. Pp. 107-115.

Gärdenfors, P. 1996. **Mental pepresentation, Conceptual Spaces and Metaphors. Synthese.** 106: 21-47.

Gärdenfors, P. 2000. **Conceptual Spaces.** London, UK: MIT Press.

Gliozzo, A. and Strapparava, C. 2009. **Semantic Domains in Computational Linguistics.** New York, N.Y.: Springer.

Gong, T. 2010. Exploring the Roles of Major Forms of cultural Transmission in Language Evolution. In **Evolution of Communication and Language in Embodied Agents.** S. Nolfi, & M. Mirolli, eds. New York: Springer. Pp. 168-175.

Guo, D.; Meng, X. and Liu, M. 2014. August. Opinion Alternating Model and Influence Factors Based on Naming Game. In **Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference.** N. J.: IEEE. Pp. 28-32.

Lakoff, G. 1987. **Women, Fire, and Dangerous Things. What Categories Reveal about the Mind**. Chicago, I.L: The University of Chicago Press.

Lenaerts, T.; Jansen, B.; Tuyls, K. and Vylder, B. D. 2005. The Evolutionary Language Game: An Orthogonal Approach. **Journal of Theoretical Biology.** 235: 566-582.

Li, B.; Chen, G. and Chow, T. W. 2013. Naming Game with Multiple Hearers. **Commnum Nonlinear Science and Numerical Simulation.** 1214-1228.

Lindh-Knuutila, T.; Honkela, T. and Lagus, K. 2006. Simulating Meaning Negotiation Using Obervational Language Games. In **Symbol Grounding and Beyond - Third International Workshop on the Emergence and Evolution of Linguistic Communication, EELC 2006**, **LNAI 4211.** P. Vogt, ed. Pp.168-179.

Lipowska, D. and Lipowiski, A. 2012. Naming Game on Adaptive Weighted Network. **Artificial Life.** 18 (3): 311-323.

Logan, R. K. 2004. The Extended Mind Model of the Origin of Language and Culture. **Proceedings of the Media Ecology Association.** 39: 149-167.

Loreto, V.; Baronchelli, A. and Puglisi, A. 2010. Mathermatical Modeling of Language games. In **Evolution of Communication and Language in Embodied Agents.** S. Nolfi and M. Mirolli, eds**.** New York, N.Y: Springer. Pp. 263-281.

Lorkiewicz, W. and Katarzyniak R. 2014. Multi-participant Interaction in Multi-Agent Naming Game. **Computational Methods in Science and Technology.** 20 (2): 59-80.

Lorkiewicz, W.; Kowalczyk, R.; Katarzyniak, R. and Vo, Q.B. 2011. On Topic Selection Strategies In Multi-Agent Naming Game. In **Proceedings of AAMAS, International Foundation forAutonomous Agents and Multiagent Systems.** Taipei, Taiwan: IFAAMS. Pp. 499-506.

Maity, S. K.; Mukherjee, A.; Tria, F. and Loreto, V. 2013. Emergence of Fast Agreement in an Overhearing Population: The Case of the Naming Game. **EPL (Europhysics Letters).** 101 (6): 68004.

McGovern, P.; Lawry, J.; Rossiter, J. and Leonards, U. 2014. Conceptual Spaces and Language Games for an Artificial Fingertip. In **Sensors, 2014, IEEE**. Pp. 332-335.

Mortenson, M. 1997. **Geometric Modeling.** 2nd ed. Hoboken, N.J.: Wiley and Sons.

Pietarinen, A.-V. 2003. Logic, Language Games and Ludics. **Acta Analytica**. 18 (30-31): 89-123.

Renfrew A. C. 1994. Dynamic Modeling in Archaeology: What, When, and Where?. In **Dynamical Modeling and The Study of Change in Archaeology.** S.E. Van Der Leeuw, ed. Edinburgh, UK.: Edinburgh University Press.

Reynolds, R. G. 1994. An Introduction to Cultural Algorithms. In **Proceedings of the Third Annual Conference on Evolutionary Programming.** Vol. 131139. San Diego, C.A. Pp. 131-139

Reynolds, R.G. 1999. Cultural Algorithms: Theory and Applications. In **New Ideas in Optimization.** D. Corne, M. Dorigo and F. Glover, eds. New York, N.Y.: McGraw-Hill. Pp. 367-378.

Roche, B.; Barnes-Holmes, D.; Barnes-Holmes, Y. and Hayes, S. C. 2002. Social Processes, Theory, Relational Frame. In **Relational Frame Theory.** Steven C. Hayes, Dermot Barnes-Holmes and Bryan Roche, eds. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Ryan, D. 2004. The Pragmatic Theory of Meaning: Negotiation by Stealth. **Language Sciences.** 26 (2): 217-229.

Steels, L. 1996. Emergent Adaptive Lexicons. In **SAB96.** P. Maes and M. Mataric, eds. Cambridge, M.A.: MIT Press.

Steels, L. 2010. Modeling the Formation of Language: Embodied Experiments. In **Evolution of Communication and Language in Embodied Agents**. N. S. and M. M., eds. New York, N.Y.: Springer. Pp. 235-262.

Stuber, A.; Hassas, S. and Mille, A. 2006. Language Game for Meaning Negotiation Between Human and Computer Agents. In **6th International Workshop, ESAW 2005, Kusadasi, Turkey, October 26-28, 2005.** Pp. 275-287.

Wittgenstein, L. 2001. **Tractatus Logico-Philosophicus.** London, U.K.: Routledge.

Wittgenstein, L. 1963. **Philosophical Investigations.** New York, N.Y.: The Macmillan Company.

Yuan, G.; Chen G. and Chan, R. H. M. 2014. **Naming Game on Networks: Let Everyone be Both Speaker and Hearer**. Retrieved from http://www.nature.com/articles/srep06149**.**

APPENDICES

Appendix A

List of Tools or Programs were Used for Construction Language Game System

1. Netbeans IDE: a Java integrated development environment.
   (https://netbeans.org/)
2. Processing : a processing of game creation and image visualizing.
   (https://www.r-project.org/)
3. R : a statistical programming.
   (https://processing.org)
4. SQLite: a standalone database.
   (https://sqlite.org/)

Appendix B

Main Class

Class 1 - Agent: Self-Organizing Map

```
import CultrualAlgorithm.CulturalA;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

public class Agent {
 public float fitness;
 boolean speaker, listener;
 int mapWidth;
 int mapHeight;
 public Map map;
 float radius;
 float timeConstant;
 float learnRate = 0.05f;
 int[] domainWeight;
 int[] coordDrawWord;
 float learnDecay;
 float getLearnDecay(){
    return this.learnDecay;
 }
 float radiusDecay;
 float getRadiusDecay(){
    return this.radiusDecay;
```

```
}
//for test
public Agent(float fitness){this.fitness = fitness;}

public Agent(int w, int h, int domainWeight) {
  //coordDrawWord = new int[2];
  fitness = 0.0f;
  speaker = false;
  listener = false;
  mapWidth = w;
  mapHeight = h;
  map = new Map(w,h);
  radius = (h + w) / 2;
  for(int i = 0; i < h; i++){
    for(int j = 0; j < w; j++) {
        map.nodes[i][j] = new Node(w,h,domainWeight);
        map.nodes[i][j].x = i;
        map.nodes[i][j].y = j;
    }//for j
  }//for
}
void initTraining(int iterations) {
  timeConstant = iterations/(float)Math.log(radius);
}
void train(int i, Space space, Node[][] node) {
  radiusDecay = radius*(float)Math.exp(-1*i/timeConstant);
  learnDecay = learnRate*(float)Math.exp(-1*i/timeConstant);
  //get best matching unit
  int[] ndxComposite = bestMatch(space);
  int x = ndxComposite[0];
  int y = ndxComposite[1];
  //scale best match and neighbors.
  for(int a = 0; a < mapHeight; a++) {
    for(int b = 0; b < mapWidth; b++) {
       //evaluate nodes between map(a,b) and prototype (x,y)
       float d = distance(node[x][y], node[a][b]);
       float influence=(float)Math.exp((-1*Math.pow(d,2))/ (2*radiusDecay*i));
       if (d < radiusDecay)
         for(int k = 0; k < space.dimension.length; k++)
           //update = learning Rate * error
           node[a][b].weight[k] += influence*learnDecay*(space.dimension[k]
                   - node[a][b].weight[k]);
    } //for b
  } // for a
} // train()
float distance(Node node1, Node node2){
       return (float)Math.sqrt((float)Math.pow((node1.x - node2.x),2) +
              (float)Math.pow((node1.y - node2.y),2));
}
public List<Node> getNeighborhood(float radius, Space space){
  Node[][] node = map.nodes;
  List<Node> neighborhood = new ArrayList();
  int[] bestMatch_coord = bestMatch(space);
  int x = bestMatch_coord[0];
  int y = bestMatch_coord[1];
  for(int a = 0; a < mapHeight; a++) {
    for(int b = 0; b < mapWidth; b++) {
       float d = distance(node[x][y], node[a][b]);
       if (d < radius){
         neighborhood.add(node[a][b]);
       }
    } //for b
  } // for a
  return neighborhood;
```

```
}

public int[] bestMatch(Space space) {
  float minDist = (float) Math.sqrt(space.dimension.length);
  int[] minIndex = new int[2];
  for (int i = 0; i < mapHeight; i++) {
    for (int j = 0; j < mapWidth; j++) {
      float tmp =weight_distance(this.map.nodes[i][j].weight, space.dimension);
      if (tmp < minDist) {
        minDist = tmp;
        minIndex[0] = i;
        minIndex[1] = j;
        this.map.nodes[i][j].space = space;
      }//if
    } //for j
  }//for i
  return minIndex;
}
public Concept getBestConcept(Space space){
    Concept best_concept = null;
    int[] bestIndex = bestMatch(space);
    List<Concept> concepts = this.map.nodes[bestIndex[0]][bestIndex[1]].concepts;
    if(concepts.size()>0)
       best_concept = concepts.stream().max(Comparator.comparing(c-
>c.belief)).get();
    return  best_concept;
}

float weight_distance(float x[], float y[]) {
   if (x.length != y.length) {
     //System.out.println("x,y length " + x.length + ":" + y.length);
     System.out.println ("Error in SOM::distance(): array length don't match");
    //System.exit(0);
   }
   float tmp = 0.0f;
   for(int i = 0; i < x.length; i++)
     tmp += Math.pow( (x[i] - y[i]), 2);
   tmp = (float)Math.sqrt(tmp);
   return tmp;
}

public int[] searchWord(String s){//search all nodes in map
  int[] indexFound = new int[]{0,0,0,0};
  //1st index ->0 = unfound, 2nd,3rd index = ij node, 4th index = index concept
  int tempBelief=0;
  for(int i=0; i<mapWidth; i++){
    for(int j=0; j<mapHeight;j++){
      for(int k=0; k< map.nodes[i][j].concepts.size(); k++){
        if (s.equals(map.nodes[i][j].concepts.get(k).word)){
          //select the most beliefest
          if(map.nodes[i][j].concepts.get(k).belief>=tempBelief){
            tempBelief = map.nodes[i][j].concepts.get(k).belief;
            indexFound[0] =1;//found
            indexFound[1] =i;//i node
            indexFound[2] =j;//j node
            indexFound[3] =k;//index of concept
          }
        }
      }
    }
  }
  return indexFound;
}
//check duplicate word
```

```
public boolean searchWordAllNode(String word){
    boolean found = false;
    int[] indexFound = this.searchWord(word);
    if(indexFound[0]==0) found = true;
    return found;
}

public int[] searchWord(String s, Node node){
   //search word in node
   int[] indexFound = new int[]{0,0,0,0};
   //1st index ->0 = unfound, 2nd,3rd index = ij node, 4th index = index concept
   int tempBelief=0;
   for(Concept concept: node.concepts){
     if(s.equals(concept.word)){
       if(tempBelief>=concept.belief){
       //if(concept.belief>=tempBelief){
         indexFound[0] = 1;
         indexFound[1] = node.x;
         indexFound[2] = node.y;
         indexFound[3] = node.concepts.indexOf(concept);
         tempBelief = concept.belief;
       }
     }
   }
  return indexFound;
}

//search best word in node
 public int[] searchWord(Node node){
   //search word in node
   int[] indexFound = new int[]{0,0,0,0};
   //1st index ->0 = unfound, 2nd,3rd index = ij node, 4th index = index concept
   int tempBelief=0;
   for(Concept concept: node.concepts){
     if(!(concept.word.equals("null"))){
       if(concept.belief>=tempBelief){
         indexFound[0] = 1;
         indexFound[1] = node.x;
         indexFound[2] = node.y;
         indexFound[3] = node.concepts.indexOf(concept);
         tempBelief = concept.belief;
       }
     }
  }
  return indexFound;
}
 public int[] searchWord(String s, List<Node> nodeNeighborhood ){
  //search word in neighborhood node
  int[] indexFound = new int[]{0,0,0,0};
  //1st index ->0 = unfound, 2nd,3rd index = ij node, 4th index = index concept
  int tempBelief=0;
  for(Node node: nodeNeighborhood){
    for(Concept concept: node.concepts){
      if(s.equals(concept.word)){
        if(concept.belief>=tempBelief){
          indexFound[0] = 1;
          indexFound[1] = node.x;
          indexFound[2] = node.y;
          indexFound[3] = node.concepts.indexOf(concept);
          tempBelief = concept.belief;
        }
      }
    }
  }
```

```
   return indexFound;
  }

  public Node findBestNodeInNeighborhoodDynmicMathWord(Space space, String
w){
     Node bestNode = null;
     float r = 0.1f;

     List<Node> neighbor = this.getNeighborhood(r, space);
     while(bestNode==null){
       if(neighbor.size()>0){
          for(Node node: neighbor){
             Concept c = node.getConceptMaxBelief();
             if(c.word.equals(w)) {
                bestNode = node;
             }
             if(bestNode!=null) break;
          }
       }
       r+=0.2f;
       if(r>2) break;
       neighbor = this.getNeighborhood(r, space);
     }
     return bestNode;
  }
  public Node findBestNodeInNeighborhoodDynmic(Space space){
     Node bestNode = null;
     float r = Config.radius_start;

     List<Node> neighbor = this.getNeighborhood(r, space);
     while(bestNode==null){
       if(neighbor.size()>0){
          try{
                  bestNode = neighbor.stream().max(Comparator.comparing(
                  n->n.getConceptMaxBelief().belief)).get();
          }
          catch(Exception e){
             bestNode = null;
          }
       }
       r+=0.2f;
       if(r>Config.radius_end) break;
       neighbor = this.getNeighborhood(r, space);
     }
     return bestNode;
  }
  public boolean selfContemplate(CulturalA cultural, Space world){
       boolean myConceptMapToCa = false;
       Concept ca_concept = cultural.ca_belief.getConcept(world);
       int[] bmu_coord = bestMatch(world);
       Node node_bmu = map.nodes[bmu_coord[0]][bmu_coord[1]];
       int[] bestWord_in_bmu = searchWord(node_bmu);

       if(bestWord_in_bmu[0]==1){
          List<Concept> concepts =
map.nodes[bestWord_in_bmu[1]][bestWord_in_bmu[2]].concepts;
          Concept self_concept = concepts.get(bestWord_in_bmu[3]);
          if(self_concept.word.equals(ca_concept)){
             myConceptMapToCa = true;
          }
       }
       return myConceptMapToCa;
   }
}//end class Som
```

Class 2 - CulturalA: Cultural Algorithm

```
public class CulturalA {
    public Belief ca_belief;
    public List<Belief> beliefs;

    //****BEGIN: VARIABLE FROM MULTI-HEARER PACKAGE
    public Space[] topics;
    public Agent[] agents;
    public Evaluation evaluation;
    //****END: VARIABLE FROM MULTI-HEARER PACKAGE
    public CulturalA(){}
    public CulturalA(Space[] topics, Agent[] agents, Evaluation evaluation){
        this.topics = topics;
        this.agents = agents;
        this.evaluation = evaluation;
        ca_belief = new Belief();
    }
    public void fitness(){

        //update fitnees of each agent
        float fitness = 0;
        for(Agent agent : agents){
            //each agent has many concept
            //compute each fitness for each concept

            //However, each topic which agent gets must
            //be the best concept (max count).

            //when they get best concept, it is ready
            //to compute objective function to get finess
            float spec = 0;
            float cohe = 0;
            spec = evaluation.specificity(agent, topics );
            cohe = evaluation.coherence(agents, topics,  agent);
            fitness += objective(spec, cohe);
            agent.fitness = fitness/agents.length;//this line just add
        }
    }
    public void initial_belief(){
        ca_belief = new Belief();
        //every things is null, it konws nothing.
    }
    public float objective(float spec, float cohe){
        return (spec + cohe)/2;
    }

    public void update_belief(){
        //1.set fitness: find the best fitness of agent
        Agent agents_max = Arrays.asList(agents)
                               .stream()
                               .max(Comparator.comparing(a->a.fitness))
                               .get();
        ca_belief.fitness = agents_max.fitness;
        //2. set ca_belief.Situation
        //get best concepts of each topic for this best agent
        //Arrays.asList(topics).forEach(t-
>this.ca_belief.situation.put(t,agents_max.getBestConcept(t)));
        for(Space topic: topics){
          Concept c = agents_max.getBestConcept(topic);//error on this line
          if(c != null){
            this.ca_belief.situation.put(topic,c);
            //System.out.println("Situation:"+ c.word);
          }
        }
        //this normative is not use if there is no reporduction
        //3. set ca_belieft.Normantive
        //3.1 find acceptable agents at Config.ca_accept% of sort best fitness
        List<Agent> agents_sort = Arrays
                        .asList(agents)
                        .stream()
                        .sorted(Comparator.comparing(a->a.fitness))
                        .collect(toList());

        int accept = Math.round(agents.length * Config.ca_accept);
        List<Agent> agents_accept = new ArrayList<>();
        for (int i = 0; i < accept; i++)
            agents_accept.add(agents_sort.get(i));

        //3.2 read best word of that topic
        List<Concept> accept_concept = new ArrayList<>();
        for(Space topic : topics){
            for(Agent ag: agents_accept){
                accept_concept.add(ag.getBestConcept(topic));
            }
            ca_belief.normative.put(topic, accept_concept);
        }
    }
    public void update_belief_newversion(){
        //a new version of paper 2
        //the new version change situation getting knowledge for direct agent to
        //get from normative knoledge



        //1.set fitness: find the best fitness of agent
        Agent agents_max = Arrays.asList(agents)
```

```
                                .stream()
                                .max(Comparator.comparing(a->a.fitness))
                                .get();
        ca_belief.fitness = agents_max.fitness;
        //System.out.println("\t\tCA beleft finess:" + ca_belief.fitness);//not OK due to
zero over time
        //2. set ca_belief.Situation
        //get best concepts of each topic for this best agent
        //Arrays.asList(topics).forEach(t-
>this.ca_belief.situation.put(t,agents_max.getBestConcept(t)));
        for(Space topic: topics){
          Concept c = agents_max.getBestConcept(topic);//error on this line
          if(c != null){
            this.ca_belief.situation.put(topic,c);
            //System.out.println("Situation:"+ c.word);
          }
        }
        //this normative is not use if there is no reporduction
        //3. set ca_belieft.Normantive
        //3.1 find acceptable agents at Config.ca_accept% of sort best fitness
        List<Agent> agents_sort = Arrays
                              .asList(agents)
                              .stream()
                              .sorted(Comparator.comparing(a->a.fitness))
                              .collect(toList());

        int accept = Math.round(agents.length * Config.ca_accept);
        List<Agent> agents_accept = new ArrayList<>();
        for (int i = 0; i < accept; i++)
           agents_accept.add(agents_sort.get(i));

        //3.2 read best word of that topic
        List<Concept> accept_concept = new ArrayList<>();
        for(Space topic : topics){
            for(Agent ag: agents_accept){
                accept_concept.add(ag.getBestConcept(topic));
            }
            ca_belief.normative.put(topic, accept_concept);
        }
    }
    public void mutation(){
        //no need to mutation
        //because this situation is not a case of optimization
    }
    public Agent[] reproduction(){
        //double[] agents = {.9, .9, .5, .9, .3, .3, .5, .2, .8, .7};
        //    .9 = 3 prob. .9/3.4 = .264  count = .264*10 = 2.6 ~ 3
        //    .5 = 2 prob. .5/3.4 = .147                  = 1.4 ~ 1
        //    .3 = 2       .3/3.4 = .088                 = 0.8 ~ 1` `
        //    .2 = 1       .2/3.4 = .058                 = 0.5 ~ 1`
        //    .8 = 1       .8/3.4 = .235                 = 2.3 ~ 2
        //    .7 = 1       .7/3.4 = .205                 = 2.0 ~ 2
        //sum = 5.99
        //List<Double> agentsList =
Arrays.stream(agents).boxed().collect(Collectors.toList());
        List<Agent> population = Arrays.asList(agents);
        System.out.print("Befor reproduction ");
        population.forEach(a -> System.out.format(" %f ", a.fitness));
        int size = agents.length;
        double sum = population
                        .stream()
                        .mapToDouble(a->a.fitness)
                        .sum();
        List<Double> prob = population
```

```
                    .stream()
                    .map(a -> a.fitness/sum)
                    .collect(Collectors.toList());
//prob.forEach(p -> System.out.print(p + "\t"));
List<Long> countList = prob.stream()
                    .map(p -> Math.round(p*size))
                    .collect(Collectors.toList());
//System.out.format("\nCount List ");
//countList.forEach(arg -> System.out.print(String.format(" %d", arg)));

Long count = prob.stream()
            .map(p ->Math.round(p*size))
            .collect(Collectors.toList())
            .stream()
            .reduce(0L,(acc, d)-> acc + d);;
//System.out.println();
//System.out.format("Count : %d  \n" , count);

while(count<size){
   double max_agent = population.indexOf((Agent)population.stream()
                  .max(Comparator.comparing(c->c.fitness))
                  .get()
                  );
   countList = prob.stream()
                    .map(p ->
                       (prob.indexOf(p) == max_agent)?
                        Math.round(p*size)+1:
                        Math.round(p*size)
                       )
                    .collect(Collectors.toList());

   count++;
```

```
        }
        List<Agent> temAgent = new ArrayList<>();
        for(int i= 0; i< countList.size(); i++){
            for(int j= 0; j< countList.get(i); j++){
                temAgent.add(agents[i]);
            }
        }
        System.out.println();
        System.out.print("After reproduction  ");
        temAgent.forEach(a->System.out.format("%f  ", a.fitness));
        Agent[] agents = temAgent.toArray(new Agent[temAgent.size()]);

        return agents;
    }
    public Agent[] evolution(){
        fitness();
        //update_belief();
        update_belief_newversion();
        mutation();
        if(Config.reproduction) return reproduction();
        else return agents;
    }
}
```

Class 3 - Evaluation

```
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;
import static java.util.Arrays.asList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class Evaluation {
    float successRate;
    float coherence;

    // List<Concept> concepts;
    int word_belief_upperBound;
    int word_belief_lowerBound;
    //Add this words variable becaurse I need only count
    //total of word that isn't identical
    //Prior I count word from total of concepts
    Set<String> words;
    public Evaluation(){
        this.successRate =0.0f;
        this.coherence = 0.0f;

        words = new HashSet();
    }

    public void addConcept(Concept concept){
        words.add(concept.word);//new modify
    }

    //specificity of a agent
    public float specificity(Agent ag, Space[] topics ){
        float spec = 0;
        String[] words = new String[topics.length];
        for(int j=0; j<topics.length; j++){
            int[] bmu = ag.bestMatch(topics[j]);
            int bestBelief = 0;//no need using
            //there are many concept in each node
            //but it speak only the best word
            //so select the best word and add it in array words[]
            for(Concept concept: ag.map.nodes[bmu[0]][bmu[1]].concepts){
                if(concept.belief >= bestBelief){
                    bestBelief = concept.belief;
                    words[j]=concept.word;//?why, because it must add all word
                    //and keep word with max belife
                }
            }
        }
        //find frequency of each word
        int[] freq = new int[words.length];
        for(int a=0; a<words.length; a++){
            for(int b=0; b< words.length; b++){
                if(words[a]==words[b])//no need use qual function since it compare itself.
                    freq[a] += 1;
                }
        }
        //sum all words
        float sum_freq = 0;
        for(int a=0; a<freq.length; a++){
            sum_freq = sum_freq + freq[a];
        }
        double nPow2 = Math.pow((double) words.length, 2);
        spec = ((float)nPow2 - sum_freq) /
                ((float)nPow2 - (float) words.length);
        return spec;
    }
    //calulate for all agents
    public float AverageSpecificity(Agent[] players, Space[] topics){
        float spec = 0;
        for(Agent player: players){
            spec += specificity(player, topics);
        }
        spec = spec/players.length;
        return spec;
    }
    /**
        Topic    1      2     3
        Agent  Conc1   Conc2 Conc3  Coh
        A1      W1*     W3*   W5*   3/3 (has 3 word star)
        A2      W2      W3*   W5*   2/3 (has 2 word star)
        A3      W1*     W3*   W5*   3/3
        A4      W1*     W4    W5*   2/3
        A5      W1*     W3*   W5*   3/3   Sum 13/3 = 4.33
                                    Averate 5 agent 4.33/5 = 0.866
        Coh     4/5  +  4/5 + 5/5 = 2.6
        Sum     2.6/ 3 concept = 0.866

    }
    */

    //for single agent //this method is used in CA
    public float coherence(Agent[] players, Space[] topics,  Agent aAgent){
        String[][] words = new String[topics.length][players.length];
        String[] wordStar= new String[topics.length];
        //1. find word;in concept,that has max frequency in all agents
        //1.1 loading topic to agent to see it
        for(int i=0; i<topics.length; i++){
            for(int j=0; j< players.length;j++){
                //agent sees the topic
                int[] bmu = players[j].bestMatch(topics[i]);
                Node bestNode = players[j].map.nodes[bmu[0]][bmu[1]];

                //agent maps to the concepts
                List<Concept> concepts = bestNode.concepts;

                //dinamic search in neighborhood
                if(concepts.size()==0 & Config.dynmicConcept){
                    //find best node
                    //bestNode = findBestNodeInNeighborhood(players[j], topics[i]);
                    bestNode = players[j].findBestNodeInNeighborhoodDynmic(topics[i]);
                    if(bestNode != null) concepts = bestNode.concepts;
                    else concepts = new ArrayList<>();
                }



                int bestBelief = 0;
                for(Concept con: concepts){
                    if(con.belief>=bestBelief){
                        bestBelief = con.belief;
                        words[i][j] = con.word;
                    }
                }
            }//end for j (players)
        }//end for i (topics)

        //for at the topic i, finding the max word using
        for(int i=0; i< players.length; i++){
            for(int j = 0; j< topics.length; j++){
            int temMax = 0;
                for (String s1 : words[j]) {
                    if(s1!=null){
                        int temp = 0;
                        for (String s2 : words[j]) {
                            if (s1.equals(s2)) {
                                temp += 1;
                            }
                            if (temp>temMax) {
                                temMax = temp;
                                wordStar[j] = s2;
                            }
                        }
                    }
                }
            }
        }

        //find wordstart commparing to wordMax[]
        int row = asList(players).indexOf(aAgent);
        int countWordStart = 0;
        for(int i=0; i< wordStar.length; i++){
            //System.out.print("Max:["+i+"]" + wordMax[i] + ": *"+ words[i][row]);
            if(words[i][row]!=null){
                if(words[i][row].equals(wordStar[i]))
                    countWordStart += 1;
            }
        }
        float coh = (float)countWordStart/(float)topics.length;
        //System.out.println("\t\t\tcohs of agent[" + row + "]:" + coh);
        return coh;
        }
        //for all agent  //use this method in game
        public float averateCoherence(Agent[] players, Space[] topics){
        float coh = 0;
        for(Agent agent: players){
                coh += coherence(players, topics, agent);
              }
              coh = coh/(float) players.length;
              return coh;
        }
        public Node findBestNodeInNeighborhood(Agent agent, Space space){
              List<Node> nodes = agent.getNeighborhood(Config.radiusFix, space);
              int bestBelief = 0;
              Node bestNode = null;
              for(Node node : nodes){
                for(Concept concept: node.concepts){
                  if(concept.belief >= bestBelief){
                    bestBelief = concept.belief;
                    bestNode = node;
                  }
                }
              }
              return bestNode;
        }
      }
```

BIOGRAPHY

NAME	Mr. Theerapol Limsatta
ACADEMIC BACKGROUND	M.A. in Translation for Education and Business from King Mongkut's University of Technology North Bangkok, Bangkok, Thailand in 2013 M.S. in Decision Support Systems from Ramkhamheang University, Bangkok, Thailand in 2006 M.S. in Information Technology from Sripathum University, Bangkok, Thailand in 2000 B.B.A. in Marketing from Kasetsart University, Bangkok, Thailand in 1994
PRESENT POSITION	Lecturer in Information Technology, Southeast Bangkok College, Department of Information Technology, Faculty of Science and Technology, Bangkok, Thailand.