Emergent Adaptive Lexicon

*Term paper for fullfiling the requirements of the course CS784
Instructors: Dr. Harish Karnik and Dr. Achala M. Raina
Authors: Nilesh Mishra and Rishabh Halakhandi
Dept. of Computer Science and Engineering, IIT Kanpur, India.

Abstract

This paper describes a simulation experiment done on the lines of Luc Steels' experiment on language evolution with software agents. In the experiment a group of agents interact with each other in a series of language games. After sufficient number of rounds we get a coherent lexicon which gives experimental proof that language is an autonomous evolving adaptive system.

1. Introduction

Language can be seen as an emergent phenomenon. Language is a mass phenomenon actualized by different agents interacting with each other with no participant having a central view or control. It could spontaneously form itself once the underlying physiological, psychological and social conditions are satisfied and could autonomously become more complex based on evolution, co-evolution, and self-organization with level formation. In AI traditionally the designer carefully designs the rules and provides the ontology. Luc Steels' work primarily tries to figure a mechanism through which language evolves on its own and becomes more complex with little outside intervention (self organizes itself).

In inductive learning (learning by examples) approach the necessary conceptualization is done by the teacher and the learning system is given positive and negative examples of the concept to be learned. It can be argued that the real intelligence in such systems lies in the teacher than the learning system. Also in many scenarios it is not always possible to get sufficient positive and negative examples. In Steels' work the necessary conceptualization is done by the agents themselves when they develop new distinctions to discriminate objects in environment. The coherence in the system is reached by language interactions. The language emerging from these series of experiments has a number of human languages like features, such as ambiguity, polysemy and synonymy. The system has positive reinforcement built into it which helps the language self organize and the system becomes more coherent as the experiment progresses. In these games as the usage of a word increases the probability the game being played with this word results into success also increases.

In the rest of the paper we present an overview of Luc Steels' work on language emergence in a group of software agents interacting with each other via exchange of lexical entities. The properties of emergent language can be studied by looking at the simulation results. In section 2 we describe language games. In section 3 we give details of the simulation and describe the experiment in detail. In sec 4 simulation results are given. Section 5 concludes with references in section 6 and definition of some keywords in section 7.

2. Language Games

Looking at language as an emergent phenomenon we find that interaction amongst various entities in the system is very important for language to emerge and self organize. Language games are a mechanism through which different actors in the

system interact with each other exchanging lexical entities (called utterances). These utterances are the viewpoint of the actor about a particular scenario. The stage consists of a speaker who generates the utterance, a listener who tries to understand what the utterance is, a topic which is the entity about which the speaker generates the utterance and finally a context which consists of other objects in the environment. The context plays an important part in terms of helping to disambiguate between different meanings that get attached with a word due to difference in viewpoints of different agents. The language game can be played in many different configurations where we can have different ways a topic is identified, what a context consists of, how and when the speaker and listener share their agreement and disagreement about the topic. Next we give detail about a particular type of language game where we choose the topic as another agent. The speaker and the listener share the topic with each other through extra lingual means e.g. pointing, etc. All the agents in the system have a number of properties which can be measured by other agents using sensors. Both the speaker and the listener try to identify the topic uniquely from the context. The speaker and the listener can differ in the way in which they identify the topic from the context and this in one way introduces ambiguity in the system. This differentiation process uses a set of features attached to different properties measured by the agents. The different steps in the process are given below.

- The speaker identifies a topic from the context and shares with the listener through pointing.
- The speaker and the listener try to come up with a distinct feature set for the topic.
- The speaker uses its distinct feature set to come up with utterance using its lexicon and if it succeeds tells it to the listener.
- The listener uses the speaker's utterance and its own lexicon to calculate a feature set say $\boldsymbol{F_1}$.
- The listener compares $\boldsymbol{F_1}$ with the distinct feature set generated by it $\boldsymbol{F_2}$ and if they are not disjoint declares the game as success.

The game fails in the following cases

- Speaker or listener does not have enough features to discriminate the topic from context. In this case the concerned agent calls its feature generation module to generate a new feature. This feature generation is random and does not necessarily means generation of a feature which can differentiate the current topic. It assumes that this new feature may help in future.
- Speaker does not have a word-feature mapping for the features in the distinct feature set. In this case the speaker calls a word generation module with a low probability (5%). The word generation module randomly generated a word. This module is called with low probability to limit the number of words floating in the agent population so that we don't have a word explosion. This can also be interpreted as a mechanism to ensure self organization.
- Listener does not have enough feature-word mapping to generate a feature set for the utterance from speaker. In this case the listener adds the concerned word for which it does not have a meaning to all the features in the distinct feature set $\boldsymbol{F_2}$ generated by it in its lexicon. This introduces ambiguity as well

as it acts as a mechanism through which the mapping of one agent's lexicon gets adopted by another agent's lexicon.

- The distinct feature set generated by the listener is different from the feature set calculated by it from the utterance using the uncover function.

The game succeeds only when the feature set generated by the listener and the feature set calculated from utterance of the speaker are overlapping. Given below are the FSM's for the speaker and listener in a language game.

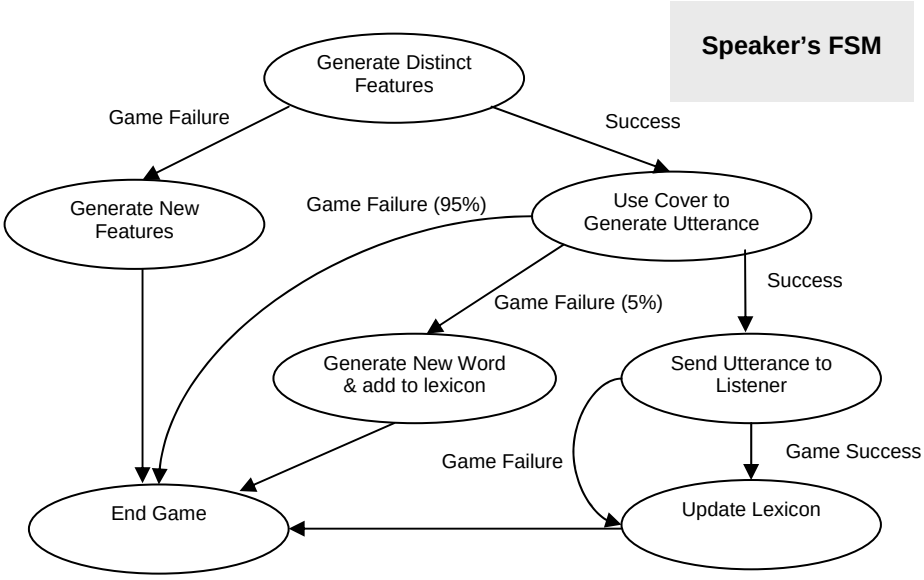

Figure 1: Finite state machine for speaker during a language game

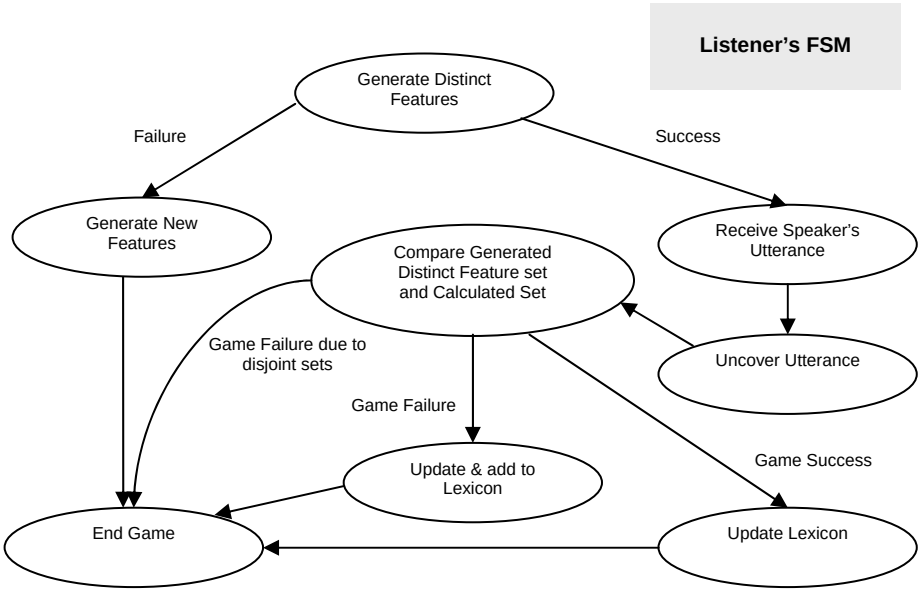

Figure 2: Finite state machine for listener during a language game

3. Experiment Details

The simulation environment primarily consists of three entities

- *The environment*: It is a passive entity which acts as a mediator and also does selection of agents for context and as topic, speaker and listener.
- *A group of agents*: The agents are the active entities in the system that plays a series of language games amongst themselves. Each agent takes the role of speaker, listener and as topic from time to time and in a random fashion.
- *A commentator*: The commentator keeps track of the system. It has access to the internal states of all the agents but no agent can query it.

The agents play series of language games amongst themselves. The experiment tests the hypothesis that language is an autonomous evolving adaptive system maintained by a group of distributed agents without central control. The system consists of an agent population (say 40 agents). In these 40 agents, at any given time only a subset of agents is active. The rest are sleeping (say 20.) At the start of the game the agents are clear slates i.e. do not have any inbuilt ontology. The active agent subset plays language games which creates individual lexicons inside each agent. With a very small probability an active agent may go to sleep or an inactive agent may come alive. The lexicon of the agent coming out of sleep is the lexicon with which it went to sleep. (This makes the system an open system). The environment acts as the mediator between the agents. It also selects the different groups (active, sleeping, context, speaker, topic and listener). An agent has an array of sensors attached with it. Each agent has a value for different sensor readings, which is initialized randomly at the start of the simulation. In the original experiment the number of sensors is limited and an agent associates a discrimination tree for each sensor to divide the sensor domain into non-overlapping, features. Each feature has an attribute and a set of feature values (range) associated with it. In the original paper the authors have divided the continuous sensor values into sub-domains using discrimination trees. Instead of dividing the sensor domain we have increased the number of sensors (original = 5, ours = 10). Every agent has a lexicon associated with it which is a mapping of feature sets and words. Each agent has a word generation module, a distinct feature discovery module and update modules for language game bookkeeping.

Distinct feature set

A distinct feature is a minimal set of features which uniquely identifies the topic from the context. The distinct feature set is found in an iterative manner where we generate different feature sets from the feature set of the topic in a combinatorial fashion. We stop at the level at which we get a distinct feature. E.g. If the distinct feature is a single feature then we calculate all the single feature distinct feature sets and do not go for distinct feature sets consisting of two or more features. The distinct feature set may consist of more then one feature when one feature cannot discriminate the topic from the context. If there exist more than one ‘feature sets’ in the distinct feature set then we need to choose only one of them for further use in the language games. The selection criterion is in the following order:

- The feature set having the least number of features used, i.e. the smallest set is preferred. This is taken care at the time of generation of feature sets when we do not go for higher order feature sets when we find a distinct feature.
- In case of equal size the set in which the feature imply the smallest number of segmentation is preferred. Thus we prefer features higher in the discrimination

tree. This ensures that the most abstract features are chosen, thus more agents can share the feature region.

- For equal depth of segmentation, we use the set for which the features have been used the most. This ensures that we develop a minimal set.

Discrimination Tree

Due to the grounding experiments where Steels uses physical robotic agents for simulation the number of sensors is limited. The goal of this mechanism is to provide an agent an adequate repertoire of features to discriminate between different agents in background. An elaborate mechanism is thus provided to generate enough features to distinguish the agent population. The mechanism starts with the association of a discrimination tree with each of the sensors. The discrimination tree divides the continuous domain into sub-domains mapping the sensory inputs to discrete categories represented as features. Whenever the agent fails to get a distinct feature set, a new feature is created. The new feature is created by randomly selecting a sensor channel and extending the discrimination tree associated with it by subdividing a distinction further. It should be noted that each sensor has a separate discrimination tree of its own inside an agent. Also the corresponding discrimination tree for a given sensor in two different agents are not correlated i.e. may be totally different in terms of branching. This algorithm is adaptive as it selects new features for future success. It is also top down as general distinctions are created before refinements are made. Also it is selectionist as distinctions are created and then subjected to pressure coming from success in discrimination.

Lexicon

The lexicon is a mapping between the different attribute-value pairs associated with an agents feature space (i.e. features generated by discrimination trees of different sensors) and words. Each lexicon is agent specific i.e. no two agents have the same lexicon. The lexicon has a bidirectional association from feature space to words and vice versa thus we have a many to many mapping. For each word entry there are pointers to different features associated with that word. Similarly for each feature entry there are pointers to different words associated with that feature. The pointer entries also contain the number of times that word/feature has been used as well as the number of time the word's/feature's use has resulted in success. Thus we maintain the status of each edge in the lexicon's mapping. These values are used for giving preference to different words in case of a conflict as well as to introduce a positive feedback loop into the system where the more successful a word is the more is its usage and vice versa. This leads to self-organization of the lexicon.

Cover and Uncover functions

The cover and uncover functions are used for generating utterances from distinct feature set and feature sets from utterances respectively. Let A be the set of active agents, D be the distinct feature set and L_a the lexicon of agent a and $F_{w,L}$ the feature set of word w in L. Then we have

- Cover function

  cover (F, L) generating set of utterances U such that

  $$\forall u \in U, \{ f \mid f \in F_{w,L} \text{ and } \quad w \in u \}$$

- Uncover function

uncover (u, L) generates a feature set *F* such that

$$F = \{ f \mid f \in F_{w,L} \text{ and } \quad w \in u \}$$

The cover function is used by the speaker and it uses the mapping from feature set to words in the lexicon. The uncover function on the other hand uses the reverse mapping i.e. the mapping from words to feature sets and is used by the listener of a language game.

Words and Utterances

The words are fixed length concatenation of alphabets. The utterances are concatenated words (separated by a separator). In the current simulation a multi word utterance has no syntactic information, i.e. word order has no significance. Multi word utterances are significant of the fact that the speaker could not find a singleton distinct feature and hence is using multiple features to distinguish topic from context.

4. Results

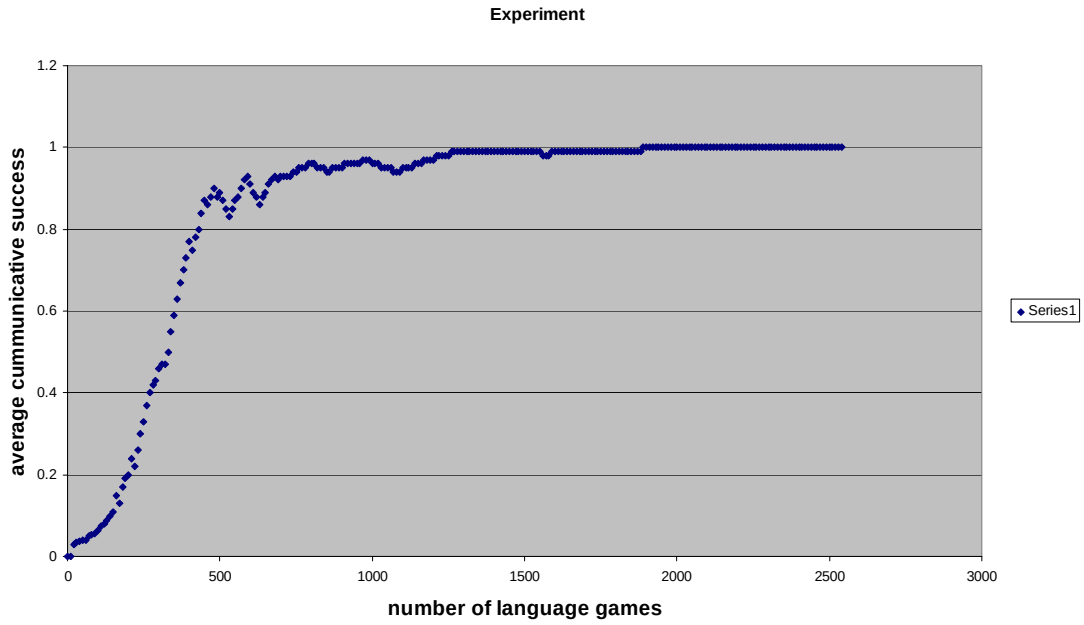

Figure: The above figure plots the formation of language. 3000 language games are shown, involving 10 agents and 10 meanings.

The above experiment was done as a java program. Our simulation results matches with Luc Steels experiment.

5. Conclusion

Steels in his series of experiments with robotic and software agents has tried to model rudimentary language systems to study the evolution of language. The significant features of these experiments are:

- Open system: The agents in the system can go in and out of the active region. Also new words and meanings are created in the system. No agent or entity has the control over the simulation environment and the complete process happens in a distributed fashion.

- Creation of new meanings: New meanings are created and introduced into the system when the current meaning set cannot describe it sufficiently.
- Only relevant meanings are lexicalized: Only those meanings are lexicalized which are used by the agents.
- Incoherence exists: The system can at times have incoherence in terms of same object having multiple representations and meanings in the system. This incoherence is resolved at the time of finer disambiguation.
- Meanings are not necessarily identifiable from context.
- Context plays a role in disambiguation of a given sentence.
- Multiple sentence words emerge in order to disambiguate single words.

The experiments highlight that language is an adaptive system which changes with change in the system (e.g. introduction of new agents). It is self-organizing in the sense that the final coherence emerges from within the system and without any outer intervention. It is also selectionist i.e. only the more successful and widely used words get domination.

The above series of experiments have been expanded later by a number of people. Some researchers have introduces learning and forgetting mechanisms to make it more real life. Additionally we can also study the model with introduction of genetic and other evolutionary algorithms.

6. References

- Steels, L. (1996), Emergent Adaptive Lexicons. In: Maes, P. (ed.) (1996) Proceedigns of the Simulation of Adaptive Behavior Conference. The MIT Press, Cambridge Ma.
- Steels, L. (1996), The spontaneous self-organization of an adaptive language. Muggleton, S. (ed.) Machine Intelligence 15. Oxford University Press, Oxford.
- Steels, L. (1996), Perceptually grounded meaning creation. ICMAS 1996, Kyoto.
- Steels, L. (1997), Constructing and sharing perceptual distinctions. van Someren, M. and G. Widmer (eds.), 1997 Proceedings of the European Conference on Machine Learning (ECML), Springer-Verlag, Berlin.
- David DeAngelis (2005), The Origins of Syntax In Visually Grounded Robotic Agents

7. Terms

- *Feature:* Concept created by the agents to distinguish/discriminate between objects and other agents present in the environment
- *Discrimination tree:* A graph which divides the continuous sensor values into subdomains leading to creation of features. The path taken gives the attribute and the node gives the value to the feature.
- *Distinguishing feature set:* A minimal set of features which will discriminate a topic from similar objects in the context.
- *Word:* A combination of alphabets.

- *Utterance:* A concatenation of words, where the word order is not important. It is the entity used for communication between agents
- *Synonymy:* More than one word mapped to a meaning (here attribute value pairs) in the agents lexicon.
- *Ambiguity:* Different agents associating different meanings to a word.
- *Polysemy:* More than one meaning attached to a word which can be narrowed down using the context. It is different from ambiguity in the sense that in case of ambiguity it is different set of agents who attach different meanings with the word while in case of polysemy it is the same agent which introduces the different meanings for the word.
- *Co-evolution:* Evolution in two or more interacting agents in which the evolutionary changes of each agent influence the evolution of the other agent.