Concepts in Artificial intelligence

Thought processing and Thought generation

Concepts and Notation

*Thought Particle* – a construct which is does not contain start symbol and end symbol and it may be mapped to a stored thought particle. A thought particle will be associated with dimensional *particle vector* where is the number of particle dimensions.

*Thought* – a construct with a starting symbol and an ending symbol which can be evaluated against a stored thought by the semantic discriminator. A thought *t* will be associated with a graph (tree?) and will be represented by an array of *thought vectors* where the array has dimensions.

A *Simple Thought* can be presented by a path from starting symbol to a sequence of thought particles with an ending symbol appended. In other words a *Simple Thought* does not have plurality of independent branches and it cannot be recursively partitioned into simple thoughts without further thought transformation which would involve the thought repository.

*Thought Repository* – a place for storing processed and mapped thoughts

*Stored Thought* – a thought which has been processed, been subjected to transformation

*Thought Transformation* – a part of the thought processing which may take place. During *thought transformation* the processed thoughts may be rearranged, coalesced or split, where each of the newly obtained thoughts is mapped against the stored thoughts accordingly.

*Particle Discriminator* – a learning module which

*Semantic Discriminator* – a learning module which

Thought Representation

– a thought particle is a piece of a thought represented by an dimensional vector where

A thought T is a construct of attached to each other thought particles represented by a directed graph in which the vertices are the thought particles and the connections between them are given by the set of connection particles where . Note that the connectivity between the particles models more subtle nuances in the semantic value of the whole thought compared to similar one.

Let us consider an example:



*I am Dimitar. My wife is Mieko. My daughters are Hanna and Emily.*

Figure 1: Possible representation

-> “***I***”

-> “***am***”

-> “***Dimitar***”

-> “***.***”

-> “***My***”

-> “***wife***”

-> “***is***”

-> “***Mieko***”

-> “***.***”

-> “***My***”

-> “***daughters***”

-> “***are***”

-> “***Hanna***”

-> “***and***”

-> “***Emily***”

-> “***.***”

We will use the function to denote the textual representation of the particle . For instance = “***I***”

Each thought particle is represented by its magnitude and direction .

Semantic Value of a Thought

A Thought is represented by its Thought Graph. There is no intrinsic semantic value associated with a Thought Graph. Instead, for a pair of thoughts we can obtain a semantic distance which is not static but dynamically evolves as more thoughts are stored in the repository and the thought discriminator internal state is modified in the process.

Building Thought Graph

Adaptive algorithm for inference of the components of the thought graph.

Start with the default representation of all particles shown on Figure 1.

Let us assume that there are already processed and analyzed thoughts:

Figure 2: Stored thought

*I is a personal pronoun. Personal pronoun is a simple substitute of proper name of a person.*

The built graph for the stored thought may look like:

One can see that , , ,

We are going to build and train semantic discriminator which will accept a thought and thought . This discriminator will produce the semantic distance between and .

Let us denote it with . We expect for a properly trained discriminator to produce small value when = and = as = “**.**”.

Coalescing of Thought Particles

Let = , =

Then =

Coalescing of Thoughts

Let = and =

Simple concatenation:

Then =

Splice operation:

=

Split operation:

=

=

If then we are going to refer to as .

Updating Semantic Discriminator through training

Naïve approach

Let us have repository with stored thoughts , , , ,