# Concepts in Artificial intelligence

## Thought processing and Thought generation

### Concepts and Notation

*Thought Particle* – a construct which 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. Compound thought particles i.e. such can be recursively partitioned in more basic thought particles are denoted square brackets e.g. p = [V1A1V2]

*Simple Thought Particle* – a Thought Particle which cannot be subdivided further into smaller thought particles.

*Connection Particle* – an artificial construct connecting two thought particles which cannot be subdivided to more elementary particles. It is used to convey semantic nuances accounting for the context in which the current thought is used. Initially, assigned some default *particle vector* with dimensions. In the thought transformation phase the connection particle c can get modified vector accounting for some inferred secondary semantic differences.

*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 path starting with start symbol and ending with end symbol. Thought will be represented by an array of *thought vectors* where each *thought vector* has dimensions. The array of thought vectors will be used to compute the *signature* of the thought . For each thought we can compute a unique *semantic signature* using *Semantic Discriminator* discussed below.

*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* cannot be recursively partitioned into more distinct simple thoughts without further thought transformation which would involve the thought repository. Simple Thought is associated with dimensional thought vector. Each element of the thought vector contains a particle vector with dimensions. Every thought can be described by an array of simple thoughts.

*Thought/Particle Vector* – condensed representation of the given particle or thought usually obtained by some sort of hashing technique resulting in multi-dimensional vector of hash values which uniquely represent the given particle or thought.

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

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

*Initial Basis* – processed thoughts and particles stored already in the thought repository at the time the input stream starts flowing.

*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* – an adaptive learning module which processes the input stream and creates thought particles by sequencing the input stream in appropriate way.

*Particle Transformer* – an adaptive learning module which re-sequences and transforms thought particles into new ones which can be mapped to stored particles or to reference particles.

*Thought Transformer* – an adaptive learning module which re-sequences and transforms thoughts into new ones which can be mapped to stored thoughts or to reference thoughts.

*Semantic Discriminator* – an adaptive learning module which computes the signature of a thought and evaluates the semantic distance between two thoughts. For the purpose it builds an internal state dynamically which it uses to evaluate the thought vector(s) of the given thought(s).

*Thought Harvester* - //TODO: finish it

*Thought Recombiner* - //TODO: finish it

*Thought Ranker* – assigns a dynamic score of each thought; this score is used by the thought transformer to resolve where to apply transformation.

*Thought Executioner* – acts on a thought marked for execution

*Small Enough Discriminator* – an adaptive learning module which finds out if a given quantity e.g. a thought or a particle is small enough

Thought Representation

We would like to have an abstract enough and memory efficient thought representation. We do not want to embed NLP constructs into the abstract thought representation.

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

A thought is a construct of attached to each other thought particles represented by a path 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 semantic nuances of the whole thought when taken in context.

Signature of thought t or particle p -

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 path*. There is no intrinsic or absolute semantic value associated with a *thought path*. Instead, there is a *thought signature* which can be computed for a given thought at a given (quantized) time. The thought signature is computed with the adaptive learning module *Semantic Discriminator*. The thought signature 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.

For a pair of thoughts, we can obtain a semantic distance which, similarly to the signature, is not static but dynamically changes as more thoughts are stored in the repository and the thought discriminator internal state is modified in the process.

Building Thought Path

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 digraph for the stored thought may look like:

One can infer that , , ,

We are going to build and train semantic discriminator which will accept an array of thoughts , , . This discriminator will produce thought signatures of each of the thoughts , , …, and the semantic distance between every pair of thoughts and – where and are in .

We expect for a properly trained semantic 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 .

Comparing Thoughts

Let be a thought represented by path and let be a thought represented by path .

Let has signature and has signature . Then and will be equivalent if and are identical or if is small enough.

Updating Semantic Discriminator through training

Naïve approach

Let us have repository with stored thoughts , , , ,

Calculating the semantic signature of a thought :

where .

Here represents the particle vector for the thought particle , .

Thought inference

*John is the father of Sam. Julie is the mother of Sam. If a person is your father and another person is your mother then you are their son. Sam is son of John and Julie.*

Thought path of the intermediate representation

By *the Laws of Attraction* we have

*<V1A1[V2A2V3]>*

Recombination of particles, laws of attraction and repulsion, inference

The laws of attraction and repulsion between particles has to be formulated in such a way which will make semantic inference possible and accurate.