**A concept network model of cognition**

By Zoltan Foris, [foris64@gmail.com](mailto:foris64@gmail.com)

We try to build a software system with thinking capabilities equivalent to human-level intelligence. Human-level means it should be able to learn just like people (often from a single example or instruction), adapt to new tasks like people (again with little new input, or even without any new input except the task definition), produce output no worse than people, possess cognitive capabilities present in children early in development (common sense, intuitive theories), be able to re-use anything it learned like people when facing a new task.

To launch this work, we selected natural language as single input and output. Full human level natural language understanding and question answering is challenging enough, and useful enough, therefore having it implemented is for sure a significant step towards the goal. On the other hand, a crucial part of the project, feeding the system with information to be learned, is by far the least resource intensive if the information is provided in natural language – especially because we aim to cover **all** domains of thinking (relevant at the age of 6).

Our current goal is to implement a system with text comprehension capabilities comparable with a 6-years-old child, to be tested by the ability of the system to learn from information provided in natural language, and answer questions in natural language. This should be benchmarked against children's performance, in all domains relevant at this age. Such system cannot work without a multi-domain common sense knowledge, therefore this is also part of the goals.

The project also aims to demonstrate the possibility to scale it to adult human level performance in natural language understanding and production.

**Summary of results**

The achievements delivered in this project so far:

1. The implementation of the concept network, which is a general-purpose knowledge representation methodology.
2. A reasoning and question-answering application written in Python. This works on top of the concept network, is capable of extracting knowledge from natural language (English) text and answering questions.
3. Initial demonstration how common sense knowledge can be added to the system using natural language English sentences, and that the system is able to understand and answer questions by learning from this input.

Current work is targeted at

* expanding the common sense knowledge base with more input and performing more tests
* performing “thought engineering”, which means we interpret the thinking process produced by the system and detect missing or malfunctioning cognitive processes, and fix or implement those.

**The concept network representation (mentalese)**

A “concept” in our terminology is roughly equivalent to a thought. Concepts are used to represent natural language text, knowledge, and inference rules. In our model, the representation of cognitive content via a network of concepts is a kind of “language of thought” or “mentalese” which is not identical to the actual language like English but is capable to represent thoughts with similar efficiency.

Technically, concepts can be produced in the following two ways:

* 1: the meaning of a word or an idiomatic phrase in English is a concept
* 2: one or more concepts connected with a *relation* also result a concept. *Relations* are a static set of possible connections between concepts.

We use a set of 20 relations, some of these are:

* C : membership in a class, IsA *A bird is a kind of animal = C(bird,animal)*
* F : feature, specification, HasProperty *Joe is tall = F(Joe,tall)*
* A : action acted by actor *Joe runs = A(Joe,run)*
* IM : implication, Causes *If Joe is adult, Joe is tall* *= IM(F(Joe,adult), F(Joe,tall))*
* NOT: negation *Joe is not tall = NOT(F(Joe,tall))* or *F(Joe,NOT(tall))*. This is also an example for ambiguity – the system is designed to manage it.

The relations we use mimic the grammar of the English language. This made it possible to develop a piece of software which translates English into the concept network representation (which we call *mentalese*). Let us take the English sentence “Money is in the bank”. In ConceptNet, for example, the representation is something like *location(money,bank).* In our mentalese, the representation is F(money, R(in, Q(bank,the))) . The relation Q is used to relate nouns with their articles or quantifiers. The relation R is used to relate a preposition with the following noun phrase. The meaning of the predicate *location* is carried by this portion of mentalese: *R(in* .

In this example we see one of the features of the mentalese representation. We only use predefined a set of 20 relations to represent any English sentence, everything else in the representation comes from the natural language form of the sentence. This is in contrast to the example of *location(money,bank)* where “*location*” is an example of thousands of symbols employed in the representation that does not come from the actual sentence.

Features of the concept network representation (mentalese):

1. Only 20 relations; on top of these only existing English words are used, no new “words” are created.
2. Meaning is carried by the combination of (1) the original English words, (2) the position of words in the hierarchy of nested concepts, and (3) the relations applied.
3. The resulting mentalese language is a rather mechanical copy of English, very much language-like. Therefore it is easy to automatically translate from English to mentalese. It is not much more difficult to translate from mentalese to English (although this is not implemented currently).
4. **We can add common sense knowledge to the system using natural language English sentences.** Even initially, forknowledge children have early in their development(intuitive physics, psychology, biology), the majority of this common sense knowledge can be fed to the system using natural English sentences. A small portion of this initial knowledge needs to be defined in mentalese directly.
5. We do not use resources like WordNet or FrameNet, because we can use plain natural language text instead.
6. Ambiguity of English is usually transferred to the mentalese expression. Disambiguation happens in the reasoning module which processes the mentalese expressions.
7. Multiple meanings of a word are multiple concepts. Therefore, it is not a *word* that is a concept, but *one meaning of a word.* The mentalese representation has two stages. In the first stage, word meaning is not disambiguated, and we take the word as the concept. This is the output of the mentalese translator, and the input to the reasoning module. One feature of the resoning module is to replace the word with the proper meaning.
8. The recursive nature of the concept network enables to represent any level of knowledge in the same consistent fashion. A sentence of 10 words is usually a compound concept of 3-5 levels of depth. Such sentences can be easily organized into a deeper compound concept, representing a paragraph, and this can be continued. The system will be able to deal with the concept of “ reasons why the US has a higher murder rate than Switzerland” using the same processes as with “Barack Obama”.
9. Concepts have a gradual truth value, currently in the range 0-4. 0 means false, 1 means might, 3 means probably, 4 means true. The sentence “Joe is probably not tall” can be represented by *NOT(F(Joe,tall))* with truth=3. This is equivalent to “Joe might be tall”, *F(Joe,tall)* with truth=1. The reasoning module performs this transformation.
10. Concepts have further dimensions that contribute to their truth value, for example a confidence level (currently 0 to 4) which expresses how solid the system’s knowledge of that concept is.

Mentalese is used throughout our implementation to store all kinds of information the thinking machine needs for its operation:

* Store input (after translation from English)
* Store knowledge
* Build models
* Define reasoning rules
* Store goals, scene descriptions, event sequences etc

In fact, the above types of information are not differentiated in our system. Input is knowledge, knowledge is model, model is rule, goals, descriptions etc. A single set of processing mechanisms processes all the above.

**Reasoning and question-answering application: “Concepter”**

We have built a reasoning and question-answering module to process the mentalese representation of the input text, called the “Concepter”. Reasoning takes mentalese as input, and produces new concepts in mentalese. Question-answering takes questions in mentalese as input, and provides answers in mentalese.

Short and long term memory

Inputs into the Concepter system are stored as mentalese concepts in two pieces of storage: short term memory (**WM** for working memory) and long term memory (**KB** for knowledge base).

* In short term memory, each input, regardless of it being a duplicate of an already stored input, is being stored separately. We use a process of coreference resolution to connect concepts which are the same. Short term memory is regularly cleared and written into long term memory.
* In long term memory, the same concepts are not stored in multiple copies, but in a single copy only, and their properties (like occurance count, truth value, confidence of truth value etc) are merged and only one instance is stored.
  + Concepts being the same on general level but referring to different specific entities are not merged because those are not the same. We can talk about dogs in general, and for this the KB has a single general “dog” concept. But if the system receives input stories about specific dogs, these will be separate concepts stored in KB.

Reasoning rules

The mentalese concepts entertained by the system represent **causal models of the world**. (The “world” is the set of inputs provided to the system.) Some concepts give rise to others in causal relationships. We call this process reasoning. The Concepter has a powerful reasoning engine that is not hard-wired but driven by rules which are also concepts.

Reasoning is driven by two types of mentalese concepts: (1) a small number of general rules, and (2) a large number of specific rules. Both rule types take the form of concepts, and in fact specific rules are often easy to express in English (for general rules the English version may seem odd or philosophical, as we are not used to verbalize these basic aspects of thinking).

An example of a general rule is the following. (I am using the word for word meaning.) If the latest input is the concept “Joe is not tall”, represented as *NOT(F(Joe,tall))*, then the system will infer the concept *F(Joe,tall)* to its WM, with a suitable truth value *.* This reasoning action is not hard wired, but initiated by detecting the general rule *IM(NOT(%1),%1)*. *%1* is a wildcard that matches any concept. This rule is saying that “from *NOT(something)* it follows that *something”.* The appropriate truth value of the implication is derived from the truth of the premises using a table that can be specific to the rule or shared with other rules.

So the form of a rule is always IM(A,B) where A are the premises and B is the implication. In general rules, only wildcards can be found, no specific word meanings. Special rules work in the same way but they include specific word meanings (they are often used to explain something about those meanings). They are subject to activation, see below.

Most general rules are added to the system directly in mentalese. This is the portion of the input that is not natural language. Special rules however are usually simply provided in English. An example for a special rule: “If something lives, then it is an animal.” → *IM(A(%1,live),C(%1,animal)).*

Activation

Reasoning uses an activation mechanism. Activation is a property on the concept (it is gradual, from 0 to 3). Only fully activated concepts and rules take part in reasoning. This is used to avoid the exponential explosion of reasoning.

The activation mechanism of the Concepter enables a simple way to make the meaning of concepts sensitive to the *context.* The set of activated concepts at any given time is the current context. For example, if the system has to decide whether a spoon of salt is “much salt”, and the current set of activated concepts has a person eating a single egg by using up this amount of salt, the system will be able to correctly imply that in this context “much salt” is true. However in the context of a salt mine, a spoon of salt is not much.

Managing ambiguity

Ambiguity is inherent to natural language. In our model we can manage ambiguity on many levels. Examples are: multiple meanings of words; multiple meanings of a natural language sentence; multiple possibilities to which an anaphor can refer; multiple meanings of a mentalese concept; multiple ways to express the same meaning in mentalese; multiple ways to translate the same mentalese to natural language. The “multiple meaning” type ambiguities are managed by allowing the evaluation of many possibilities in parallel, and selecting the best ones (this is currently implemented in the Concepter for anaphor resolution). The “multiple ways” type ambiguities are managed currently by allowing the reasoning of several concepts with the same meaning (restrained by activation).

Exceptions, contradictions

Natural language texts are full of contradictions, which come in many forms. Two cases are currently managed in the Concepter system.

We can handle exceptions – concepts that contradict to some general knowledge. For example, the system knows that “people can talk”. Later the system learns that “people less than 1 year old usually cannot talk”. This is resolved by noting which concept is a special case of a more general concept. “people less than 1 year old” is more special than “people”, therefore reasoning with it will prohibit the reasoning with the more general concept.

The system also encounters contradictions or reinforcements not involving a generality-specificity relation. In such cases some properties of the concept (like truth value) get adjusted.

Inductive reasoning

Rules are usable to support inductive reasoning. For example, when reading about a specific cat and learning that it has a tail, the system can induce that cats in general have tails.

However more work is needed to make this really useful, and to introduce further forms of induction, for example analogical inference. This is subject to future work.

**Future work**

In order to achieve the goal of domain independent language understanding on 6-years-old level, we must provide common sense knowledge for the system. This needs to be manually selected from existing pieces of text, or even written, as there is no organized resource available for domain independent children level common sense. The effort can be significant (or even too big for a research project), however we have written the first dozens of pages and this input is already usable to demonstrate how the above functionality works.