Brief recap

October 2017

Being in the current situation, trying to tackle the problem of text entailment and more particularly, given a pair of sentences $t = (a_0, ..., a_k)$ and $h = (b_0, ..., b_n)$ give with the highest possible degree of accuracy a proxy of whether h follows from t or not. In such scenario it might be handy to use the content described in the first chapters of 'Representation and Inference for Natural Language'.

For the sake of simplicity we assume that the initial pair of sentences has already been treated and factorized into their respective decomposition. We can regard this initial state either as an embedding of the original space of finite sentences to the space of syntactical trees or as a batch normalization to ease the further treatment and its explainability.
Lambda calculus is a tool for controlling the process of making substitutions. With its help, we will be able to describe how semantic representation should be built. The big idea behind it is to enable us with the possibility to set variables in any place of the expression just as if they where placeholders and later evaluate the expressions querying the desired value just once.
Across the book several examples are provided to ease the comprehension of the content explained, a particular case of the afore-mentioned representations is given the sentence "Vincent likes Mia", we stress the fact that it is not only a string of words but rather a hierarchical structure and the representation that best projects this information is LIKE(VINCENT,MIA).

And this particular type of representation gives rise to the compositionality which is one of the key concepts of contemporary semantic theory, that is, given a structured hierarchical structure we intend to capture the semantical content with a short expression just like in the case above.

The structure presented is the following: A problem and a solution are provided and then the author tries to develop a model by trial and error according to which the model will have as an input the initial problem and hopefully as an output the desired solution. The first lesson we learn is that to build representations, we need to work with incomplete first order formulas, and we need a way of manipulating the missing information in our context we can associate to them a prolog variable. Or in the case of lambda calculus, the λ simply stands for an operator that marks missing information by binding variables λx.MAN(x) here λ binds the number of occurrences of MAN, nevertheless this seemingly simple notation allows for a much more complex and enriched computing structure.

As a sample example lets examine what's involved in building the semantic representation for 'every boxer walks' using lambdas. every: $P.\lambda Q.\forall x(P@x \rightarrow Q@x)$. boxer: $\lambda y.BOXER(y)$, walks: $\lambda x.WALK(x)$. And the beauty of this hole concept is that by treating the sentences recursively we allow them to have very rich substructures since for a given sentence a tone of information can be derived from every one of its particular components and not only the resulting expression should be taken accountable.

Many more obstacles will be posed along the way since not all the treated sentences will be as simple as the ones presented so far. If we regard the sub parts of a sentence as a structure by itself there are many non equivalent ways to expand them and compound them together so that we have a general structure, this non uniqueness leads to some uncertainty and under specified representations is the result. (I refer to deepen this knowledge Montague's approach) in which instead of directly combining syntactic entities with the quantifying noun phrase we are permitted to choose an indexed pronoun and to combine them together. The pronouns play the role of placeholders.

Even if this approach may be regarded as deprecated seeing some of the tools nowadays available (refer to the literature in github) such as machine learning and neural networks as a whole, this is sort of very costly in the beginning, fine tuning is necessary and a lot of choices have to be made instead of letting an algorithm find a good fit for the already seen train cases but explainability is provided. The resulting model does not act like a black box type of algo but rathe as a stacked set of smaller models to provide a final answer.