A feasible path to a Turing machine

The recent progress in AI has been phenomenal. This has been made possible by the advent of faster processing units and better algorithms. Despite this rise in the state of AI, there is a lament that most of the work has been to apply the new advances to a narrow scope and the original intention of the "founding fathers", the creation of a machine that has human level abilities in processing information, has not attracted much in the way of attention. This paper aims to propose a path that could lead to the creation of a so call AGI. An artificially intelligent machine whose intelligence equals that of the average human in all areas that humans display intelligence. I will focus on the field of Natural Language Processing, as i believe that that is where most of the work is needed.

The problem with the traditional approaches to AI is that inference is largely stochastic, while in humans this may also be true, but this stochasticness is constrained by rules such as can be found in the theories of logical transformations. Using a purely unconstrained probability function to predict the presence of a particular object in an image or to infer a particular fact from given propositions, while this may work somewhat, will not be sufficient to get us to the greatest level of performance. Though some NN designs such as convolutional networks address this partly there is still a long way to go. The other problem, while not a mortal one stems from the approach most researchers have taken. That is to model the brain in order to do what the brain can do. I call this a solution by proxy. Little has been done to approach the problem directly. The direct approach would take into account the range of activity that the brain can perform and then create a structure that can duplicate that activity whatever that structure actually is. That would also be a form of solution by proxy, because we aim to get the activity from this new hypothetical structure just as Deep learning theorist get it from the brain. The difference is that we allow the form of the activity (be it conversational ability) to induce a model of a structure that can support that form. Another consideration is that while we are not the only ones with brains in the animal kingdom, our brains are developed enough to permit complex tasks. Given that our brains are largely an extrapolation of those of lesser creatures we would begin to wonder if we would want to keep extrapolating the abilities of the brain, because that might make us susceptible to a blind spot where activities that would or should be defined as creative and intelligent will never be possible using a deep learning algorithm because it can only follow the path of the extrapolation and not deviate from it. In short I am saying the architecture of the human brain may not allow it to conceive certain things, and with deep learning, we are recreating this flaw.

In this paper we will outline a NLP system that is based largely in graph theory, and together with techniques found in linear algebra we will be able to model the rules of logic. Inference from given data in natural language format will then be done by creating a mapping between the premises and the conclusion. During training the vector that is responsible for taking us from the space that the premise occupies to the space that the conclusion occupies will represent the particular logical rule used.Also training involves determining a particular vector for the job.

Before we begin we would like to consider some key features of what we want to model, before we create the structure that will produce them. Consider the statements below.

1. If I wake up late I will miss the bus and I woke up late.

2. I miss the bus.

Statement 2 follows from statement 1. The description of this relationship is called Modus ponens, which is a particular logical rule. There are many examples of statements that relate in this way using modus ponens. Now statement 1 and 2 are related. And to be related in mathematical terms is an ordered n_tuplet. This particular 2-tuplet could be represented as the ordered pair (1,2).Since order matters it is clear that to be related you have to stand in relation. So we can imagine these statements to be embedded in some space, standing in relation to each other by their coordinates in that space.

That relation that they stand in is the relation of modus ponens. And the vector that takes us from the point where 1 is to the point where 2 is represents modus ponens. Consider the next two statements.

3. If I don't eat I will die and I don't eat.

4. I die.

The move from 3 to 4 is also an example of modus ponens "in action" so we could place these two statements into the same space as 1 and 2. They would occupy different positions in the space, but because both (1,2) and (3,4) "stand in relation" using the same modus ponens logical rule (i.e. the same way) the vector that takes us from 1 to 2 will have the same gradient and magnitude (i.e. will be equivalent to) the vector from 3 to 4.

So far we have expressed 1,2 ,3 and 4 as points within the space we embedded them in. This was not entirely accurate because we know that for example both 1 and 3 contain the word "if" in their expression. So we need to be able to illustrate this in our space *without* adjusting the vector. Or if we adjust the vector we would like all the logical pairs that interact using modus ponens to have the same vector. That means that we need to place all the words in the language we are using into that space and then define how we group them to represent sentences ,so we can have a vector between one group of

words (the premise) to another the conclusion , in such a way that all tuples that represent a modus ponens transformation use the exact same vector.

The solution I propose involves defining a force similar to gravity into this space then connecting all of the words in the space with vertices. These vertices may have different lengths and this graph that we have created may not be symmetrical.If we give some mass to any node/word, the center of gravity of this graph in this hypothetical world will shift and position itself somewhere in the space dependant on the position of the node that had its mass increased.

We now have the tools necessary to group words together. When we are given a premise we increase the mass of the words in the premise. This will give us one point which will be the new center of gravity of the graph. We note its position. Then we reset the graph before giving mass to the words in the conclusion and noting where the new center of mass is. We then have two points (the two centers of gravity of the premise and the conclusion) that we can connect with a single vector.

Were we to have another training example we would again generate two unique centers of gravity that we can connect with a vector.

Depending on the location of each of the words in the graph, the vector will be different, as there will be different centers of mass for both premise and conclusion. Our task becomes then having a particular arrangement of nodes and vertices, such that all examples of modus ponens “in action” have the same vector. Having the same vector means that they all have the same gradient (are parallel) and also have the same magnitude. Training will involve per mutating the position of the words in the space until this happens. As we per mutate the words, the vectors for each example will change. What we need to do is reach a point where they are all parallel and have the same magnitude. During operation we will use this trained graph, increase the mass of the words in the given premise, determine its new center of mass, then placing the learnt vector for modus ponens with its start point at the position of that center of mass. The tip of the vector would then represent the center of mass of the conclusion . A method for determining which words in the graph would give that center of mass is beyond the scope of this paper and involves an additional algorithm.

In order to train the system we need to initialize the position of the words in the graph randomly. Then place the first training example and create its vector from the two centers of mass. We will then move the nodes of the initialized graph in straight line in different directions. We will need to have fixed boundaries where the nodes cannot cross. If a node meets a boundary it will be deflected. It is important that we model the exact same movement of words exact to every detail of position of the words with respect to time. This is because we will give each training example the exact same series of movements in the graph when it comes its turn.

We then create a 3 Dimensional Cartesian coordinate system with the axis ;time , magnitude and gradient. A curve will then be drawn using the information from the vector in each of the training examples. As the nodes move with time, the vector for the example will change its gradient and magnitude. We will plot this on the 3D Cartesian coordinate system.

We will perform the exact same procedure for each of the training examples. What we need is a point on the 3D coordinate system where all the lines drawn on it during the training period to meet at one point. We will then observe at what time this occurred and the configuration of the nodes/words on the graph when this happened. This will be the final orientation of the graph.

In order to speed this up we would need to place the words in a relatively restricted part of the space they are in, using the boundaries. That is because we are only concerned with the *relative* position of the words, relative to each other. And for each configuration of nodes there is a symmetrical one that is just proportionately smaller. The vectors will also be smaller but we do not need a particular size for the vectors, just that they must all be the same size (whatever size that is) and parallel (whateve gradient they have).

To develop this further we would have a separate graph for each of the logical rules we wish to model, and train for a vector the same way.

So far we are still far from having created a Turing machine. That is because we may need to have a long process when deriving information that uses many logical rules and the hypothesis together with conclusions from previous steps. Once we have trained for the rules, it still requires human intervention to plan on the steps need to completely prove a particular fact using many rules and to select the relevant hypothesis from a database. The proposed solution to this will not require any human intervention.

We could have used just one graph to model all logical rules. If we had ,what would we say about the set of all legible statements within this graph? It should be clear that there will be vectors from all of the statements that interact using modus ponens that will be the same, and vectors from say all the statements that use the hypothetical syllogism logical rule that are the same. In short this all inclusive graph, due to the way each of the individual words in it “stands in relation” to each other (i.e. the orientation of the graph), all possible legible statements also stand in relation to each other.

To use this insight we would train a new graph system to be used alongside in a particular way. We would do it such that each of the hypothesis (statements) used in a particular step of a proof have vectors that meet at their tips. These vectors would be trained so that all statements that have the potential to interact using modus ponens will have vectors that meet at their tips and have a particular angle. So now we a looking for a valid angle as well as vectors that meet, for the statements that are

related by the logical rule that we are modeling. Note that each vector belongs to a particular statement and not more than one. The construct of all the vectors meeting at a single point with a particular angle between them represents that complex of statements.

Then as a document is read we feed in each statement in it into this new graph (during operation) and determine the vector for it. The value of the vector is then associated with that sentence in the database. As it reads, statements with vectors which show that they would meet at the tip and complete the full requirements of angle as well. Will be shuttled to the relevant graph system to derive a conclusion from it. Then this conclusion is place into the new graph to determine its vector and stored in the database as well. When this new conclusions vector meets with another statement in the database that is related by some logical rule, the process is repeated.

This is just the tip of the iceberg concerning where we can go with this. Additional work needed would be done to develop a method for the combined system of all these graph systems to present the information it has deduced using the stated methods. This could be done in a conversation as Turing envisioned or simply as a question answering system.