>SOLVE ZORK: Teaching An AI To Play Parser IF

Recent advances in artificial intelligence allow us to solve problems and design technologies that not so long ago were science fiction. We ask selfishly: what does it mean for parser IF?

Deep learning

Throughout most of the history of computing, to get a computer to solve a problem for you, you first had to think about it really hard, then program the computer to perform some steps in the right order. This worked very well — it pushed humans to come up with creative concepts, do some theoretical work, and ultimately advance science.

But think about it: is this how you were raised? Are our children programmed, perfectly obedient and only as intelligent as we are? Do we even understand what is in their brains? This is the paradigm shift brought about by artificial intelligence: create structures to hold information and a great number of parameters (basically, a brain!), and a way to have these parameters adapt locally and internally as a function of external feedback.

This is the idea behind *neural networks*, proposed in the 1980s: neurons hold a small quantity of information and interact with each other, reconfiguring pathways and weight given to each element to get better. The main problem with neural networks is the processing power involved in making a computer that's at least decent in accomplishing its task: you have to provide millions of examples, with enough diversity that the neural network will know how to deal with the full range of situations. Then for each of these examples, you have to actually perform the simulation of each neuron's behavior and their interactions and compute the adjustment in the neurons as a function of the result. This limits neural networks' efficiency, and they were abandoned by the '90s as simpler methods gave results that were just as good. Enter deep learning, and the neural networks make their comeback in style.

Now might be a good moment for a caveat: I am very far from being an expert in these matters, but I was lucky enough to be able to attend a conference by Yann LeCun, one of deep learning's best researchers and head of AI at Facebook Research, and the whole thing blew my mind! A lot of things were explained simply, and there were so many examples, including some that inspired this article. So I'll try to explain this as best as I can, and refer you to the slides if you want to know more.

The "deep" in "deep learning" stands for "a large number of processing layers". The idea is to have a lot of specialized (and trainable) components that perform specific operations, such as filtering, on part of the input; higher-up layers aggregate results from these layers, which makes them able to look at higher-level features of the input. Just like in the visual cortex: simple cells in the eye get a few pixels of the image, more complex cells aggregate these small pieces of information to get features of an image, then more complex cells recognize some features as part of an object, etc!

This visual analogy is actually precisely why deep learning has been so successful in image recognition and classification, achieving incredibly low error rates.

Deep learning is very recent — no more than five or ten years old. It still requires a lot of computational power, but several racks of current graphics cards are enough to train efficient silicon brains. The results are astonishing: deep learning is already a reality, hiding behind Siri and Google Now, automatic picture tagging on Facebook, and the victory of AlphaGo against Lee Sedol; in the future, they'll be in self-driving cars, chatbots, and MRI machines.

Building a computer with common sense

Interestingly, deep learning methods are also applied to natural language problems, which are one of the most difficult classes of problems that we may want to teach computers how to solve. This includes problems that human brains solve daily: parsing sentences, linking words to abstract concepts, performing small logical deductions, perform abstract reasoning, and even translation! This is the heart of human communication, reasoning, and consciousness, much more than playing Go or identifying a picture of a banana; for this reason, Alan Turing proposed that the ability to chat with a human so well that the human doesn't know if she is speaking to a computer (the famous "Turing test") is the definitive sign that computers are as smart as humans.

Hence, giving a computer "common sense" is still hard and unsolved. How do you teach a computer to know that if Jack has a glass of milk and went to the office, the glass of milk is also in the office? How do you teach it that if you close the oven, the food is still in the oven? That water does not burn? That people cry when they are sad?

We'd need a way to show the system thousands of situations that highlight properties of the physical world; thousands of different objects and settings, so that it learns what usually goes with what; thousands of occurrences of objects interacting with each other. And maybe give it a way to interact with such objects, so that it experiments and gets better at learning physical properties in an unsupervised way, just as AlphaGo got better by playing against itself thousands of times. But if you drop them in the real world, it's going to take a lot of time and a lot of effort (as well as a lot of broken things): you have to learn how to operate arms, vision, etc. A 3D simulation, maybe? It's been done: researchers have used the Unreal Engine to teach a computer how physics worked, so that the computer ended up predicting that an object was going to keep falling until it hits the ground. But this requires image processing, even real-time processing abilities.
But *why don't we make the computer play parser games?* This is learning through play, using the textual nature of parser games to teach the system natural language, and common sense reasoning through puzzles; and there are literally thousands of games on the IF Archive, so lots of material to learn from!

This is not a totally original idea. Myke, a user on intfiction.org, recently opened an intfiction.org thread to ask for help with his research, which consists of exactly that: using parser IF to teach an AI about language understanding and physical properties. As he notes, "the dream scenario would be to let the system train on a set of IF games and then test it on games that it has never seen before and see it perform well." (We wish him all the luck in his research and hope he'll show us his results!)

Furthermore, as he notes, there are least two recent papers that have looked at text adventures as a way to train an AI to understand text and physical properties. However, these works train a neural network to get better and better at a game: the resulting neural network does not learn how to play games in general. The first one, by MIT’s Karthik Narasimhan, Tejas D Kulkarni and Regina Barzilay, attempts to teach a system how to play a simple parser game (called a MUD in the paper) in a house (sample quests: you are hungry but not sleepy, and there is a bed in the bedroom and an apple in the kitchen), then a larger game in a fantasy setting; the system eventually manages to complete the game on each try, once it has completed the game 50 times. The paper attempts to see if any of the gained knowledge could be “transferred” to other games, but their method is baby steps, as they only show that if you move the bedroom and the kitchen (but keep the same objects and descriptions), the system learns how to solve that game faster. The second, by Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng and Mari Ostendorf, uses choice/hypertext games where the choices are randomized (to prevent the machine from simply remembering the number of the options); the computer then gets better and better, and seems to also gain an understanding of the situations, as the second part of their experiment (replace actions with paraphrased descriptions of the actions and see if the system is still able to get a high score) shows. All this is preliminary work, but rather exciting nonetheless!

What and how a computer would learn from parser IF

Let’s step back a little bit and break down what this all might mean, from a text adventure author/aficionado perspective. In particular, we’ll take a closer look at different aspects of text adventures and player behaviors, and see how they might apply to an AI able to play parser games. This is also helpful to take apart mechanisms of parser IF and see what they demand (and how hard they can be) for a completely uninitiated user.

Input

First of all, we should ask: is this even doable? The only way to know for sure is to attempt to build it. Deep learning is an interesting and powerful technique, but there are a lot of challenges. For instance, the syntax of commands: normally these consist of a verb followed by an object, but some verbs have prepositions (e.g. ask X about Y, put X on Y), which complicates matters because the English syntax is not the same for every verb. And for that matter, doesn’t this imply that the system should first figure out what’s a verb? Or even what’s a valid word?

From various attempts at coding AIs that can play video games, the process might go like this: a completely newborn system would probably ignore the command line for a long while, then figure out that typing is an interesting idea, then type absolute garbage for a long while until it gets the idea of typing a command starting with “x ”, then go on an examine binge until it finds a word that matches, hopefully learning that words that are in the text are more likely to get responses. But how does it figure out “take” when most games don’t even mention it? An interesting alternative would be to bootstrap this process by enforcing that the system can only type a word from a selection of words – for instance, all the standard verbs, plus any word seen in the output of the game. Then the system would “just” have to figure out that >swim never produces anything useful, as opposed to

>examine or >north.

Even with just that, we are talking hundreds of verbs, hundreds of nouns and countless words from the game's output, while previous attempts to make computers play video games focused on console games with half a dozen choices at each frame, instead of tens of thousands. The large search space simply means that more computations would be needed to figure it out. But in theory the computer will figure it out *eventually,* given enough time, processing power, and probably memory – a non-trivial task.

Reinforcement

But let's carry on and assume we have a system that knows it should type words. We then have the question of the "objective function," of the feedback that lets the system modify itself: what kind of things should the system pursue, and what things would let it know it's on the right track? First of all, it seems like we should restrict ourselves to playing games with at least one preferable end state, or our poor machine will get confused trying to figure out what it needs to do to finish the game. Exit *Aisle* and *Galatea* (wouldn't a computer talking to Galatea be the most meta thing, though?) However, it would be very interesting to train a system fully, then make it play these games: will the system loop infinitely, looking for more endings, or will it think "yup, that one, that one is the best ending" and stop? Which will it choose?

Back in the day, most games had a scoring system that would reward you with points for quests, subgoals, secrets, etc. This kind of feedback seems like a natural candidate for our AI: the higher your score, the closer you are to the end of the game, and maximizing your score means you completely beat the game. Furthermore, perhaps timed sections will teach the AI that time should not be wasted and the number of turns should be kept to a minimum. But eventually, the system will have to play modern games, perhaps with puzzles but no automatic score reward; how would the system react to this kind of game? Obviously it'll continue playing, but it will probably walk around aimlessly and unfocused for a while. Then it could figure out other, more subtle encouraging signs: managing to take an object, reaching a previously unreachable area, a NPC that says "thank you" — at which point the AI would be starting to think like a player!

On a side note, one cannot talk about things that signal that you're on the right or wrong track without talking about the cruelty scale. Merciful and Polite games seem to be the best for this system, as once it has figured out that nothing can be done when dead except restore or undo, it'll probably have learned that dying is bad. Tough games seem a bit more complicated: the game will give warnings before the act, but if the system doesn't register them, it will perform the action, then keep on playing and remain locked out of victory for thousands of turns. Nasty and Cruel are even worse, but again, they are a challenge even for human players! The mere problem of detecting that you're stuck seems like a very hard problem: how do you know that the game will never let you win, how do you know that there's not a small non-obvious action that you have forgotten to do? This is often solved by glancing at a walkthrough – which obviously is not applicable in our case. So this particular problem seems to be particularly prickly.

Objects

One thing we have not mentioned yet is the relationship with objects, which is crucial in parser games. We humans generally acquire object permanence, i.e. the mental process which says that objects have an existence even if they are not currently seen, before we are two years old — rephrased differently, this is a skill that takes humans two years to learn! Because traditional parser games are primarily interested in object manipulation, this seems like the perfect way to teach the concept to a machine, and this is probably useful in a lot of applications which deal with giving the machine something that resembles common sense.

This will presumably mean that, for a long time, the system may go to the room where the troll guarding the bridge is asking for a fish, look for a fish in the room, go to the next room, see a fish, and fail to connect these pieces of information as the troll is not in view anymore. On a more basic level, the concept of inventory is one that should be very hard to grasp for the machine! I mean, think about it: there's the object you need, but you typed "take" and now the object is gone. This is clearly not good, right? But eventually, the game should figure out that typing inventory will make the object appear – and even better, that if you take an object in a room and then move, the object is still in your inventory.

This should be a major breakthrough for the system, and besides, it's necessary to solve puzzles (or even figuring out that taking objects is a good thing and not a bad thing!). No doubt that learning this skill will be a large advancement in giving a computer common sense.

Puzzles

It is now time for our system to attempt to solve puzzles. Much has been said about the design of puzzles in adventure games elsewhere, and it is interesting to think about the different pieces of advice that have been given and recast them in our context of an AI learning how to play parser games.

For now, we just wish to highlight a few things on fairness. Puzzles are often qualified as "fair" or "unfair"; what would happen if an AI stumbled on an unfair puzzle? A puzzle can be unfair for a lot of different reasons and may stump our system in numerous ways. The puzzles where you are not warned at all before dying, or have to die and retry, may be solved easily by the system (once the command "restore" is known), as a computer would presumably not get as frustrated as a real player would. The puzzles where no hints are given are a bit harder to solve, but ultimately, if the machine is smart enough not to loop and try the same cycle of actions over and over again, it may well stumble on the right solution given enough time. The same applies to puzzles that require external knowledge or rely on in-jokes: the "hint" may as well not exist. One of the trickiest types of puzzles is the puzzle that requires a very specific formulation, a specific verb, or even those for which a perfectly sensible solution should work but doesn't. These puzzles are notoriously frustrating for humans, who eventually turn to a walkthrough, and the question of how a machine would solve it is interesting. It may require the machine to learn synonyms of a verb or of a noun (which is a very abstract and very high-level concept), and enough stubbornness to keep going down a path while metaphorically screaming "this should work!" Hence — as it is for humans — it is probably best to avoid unfair puzzles.

Before talking about fair puzzles, let's deal with the question of mazes, as these are inevitable when playing a large number of parser games. The standard maze is the one that only needs to be mapped; this implies that the AI system figures out what mapping is (unless the strategy of drawing a map has already been found just to be able to move around) and has enough memory to draw one. We then increase the difficulty by having non-reciprocating room exits; then perhaps some random events, exits or rooms, which increase the difficulty quite a bit. Then, imagine the system's confusion when encountering "a maze of twisty little passages all alike": it has to realize that these are indeed different rooms (and not just the same room), and find a way to distinguish them, rediscovering the standard "drop objects" technique: this strikes me as quite a feat of logical reasoning. Then there is the great number of mazes that require the player to realize that this is not a standard maze and that there is a trick. These require even more logical deduction, and presumably great maze experience, for an AI to be able to figure this out. So mazes themselves seem to be a very complex topic, one whose resolution would be reasonably considered to be an advancement of science – can you imagine having an AI crafted specially to solve mazes for you?

Let's assume we stick to fair puzzles, then. Discounting clues given to the player, one can theorize that fair puzzles are the ones that rely on common knowledge or common sense, and that make sense once solved. This is a prime example, and maybe the main reason, why parser games are an amazing possibility to teach AI systems common sense: by learning through play and experimentation, the system will notice patterns that correspond exactly to common sense, and will learn and perhaps reinforce some new things, sometimes by blind luck. Things may start slow, by learning how to solve "measure 5L of water with 2 cups of capacity…" puzzles a thousand times, but presumably also by learning how to retrieve a key with a bobby pin and a newspaper. Things can then progress to more advanced concepts, like light puzzles, or the use of mirrors: games which rely heavily on simulationist aspects, like the *Zork* games, can be great to learn about concepts like light sources, liquids, burnable objects, or even ropes, as they allow a lot of experimentation and deal with fundamentals of our physics. The caveat is that the game has to have very few bugs, so that the AI is not misled into thinking that these bugs also happen in reality. Increasing the difficulty, one could think of associations of such concepts, as in the solution to the maze in *Photopia*; and finally, more complex cultural concepts (setting off the fire alarm makes people vacate the premises). Ultimately, there are so many games, and so many puzzles, that the sum of the total common sense contained in the IF Archive is probably enough to make a really smart AI!

Implementation

We wrap this up by talking about the problem of game implementation. What happens when the game is full of bugs, typos, is finicky about grammar or formulations, or didn't implement any of the obvious solutions for a puzzle? Or just when the game is just not very thoroughly implemented? The human player then thinks "this game is buggy, this sucks — but I should be more stubborn and more careful" – i.e. maybe drop ideas less quickly than with a game which is otherwise superbly implemented. If this kind of "meta-gaming" is not done, much frustration can occur: you can search blindly in the dark for a long time, only to find out that the thing you tried in the first place was actually the right solution, but needed a minor adjustment to work!

Can an AI system reach this level of awareness? This reasoning is very meta, relying on the human aspect of game-making: "this game is not reality, it is something that a human has made, and I know that planning every single answer is not possible (or: this human was not very thorough), so I'll suspend my disbelief and keep playing but keep it in the back of my mind to advance". Two things can then happen. The first one is that an AI never realizes this, and keeps being tripped up by this: it

drops solutions too quickly and never tries them again, doesn’t manage to solve a puzzle in a game because it learned in another game that this solution doesn’t work, etc. Differences in implementation create differences in the simulated game, and from the point of view of the AI, inconsistencies in the common sense; it would have to be robust enough to recover from these, which is very hard to imagine as it is supposed to incorporate feedback to its very core.
The second possibility would be that the system is indeed able to perform this logical jump, and realize that these are only simulations imperfectly created by humans (whose names are given in the first paragraph of the game). Presumably, the AI would adapt its play in function of what it perceives the style or quality of the game to be, or even the author; but fundamentally, this possibility means we were able to create an AI that has now realized it has been living in simulations this whole time. Science fiction then dictates a bad ending for humans – so please, for the love of god, polish your games!