Creating Intelligence is a Problem of Philosophy, and Programming 3

By James Lewis
July 15, 2021

This is the third in a series of short papers discussing why creating intelligence is more likely today than ever before, and not because of deep learning. Programmers have other options for developing intelligence. But before I continue, I need to answer some important questions, such as:

Why should we create intelligence? Considering the disappointments investors and technologists are experiencing with practical AI solutions, it's a timely question. There can be only one answer. We create intelligence for our pleasure.

But what is this pleasure and where do we find it in our solution? It turns out there's a book that answers that question, "The Pleasure of the Text" by Roland Barthes. Barthes had a unique talent for de-mystifying our relationships with books, photographs, movies, news, fashion. To continue this project without understanding his work would be like developing software for an enterprise with no information from the people who understand the enterprise best. In terms of culture and communication Barthes is the subject matter expert.

Here are a few points Barthes brings up:

- Human-created intelligence must demonstrate *desire* for person it interacts with; it must convince that person that he or she is wanted. There are various ways of doing this, of which anyone in a relationship is aware, or needs to be.
- Barthes also refers to "cutting" or the cutting and fitting together of two disparate things. (May be referred to as antithesis.) An example of this would be a movie in which a bank teller and philosophy student meet.
- Also, nothing should be revealed all at once. We should only get glimpses of what we want to see, intermittently. That way we can enjoy seeing things being revealed.

And now comes the question of how. How do we provide the pleasure? For the intelligence we create, the intelligence must decode what its interlocutor says, successfully, or not (even we humans often have a problem with this) and then project a thought (sentence) into the conversation.

And these are helpful technologies we have today for doing so:

Object-orientated and functional, or declarative, programming

Sentences are, or course, composed of words, words with different properties that must be sorted through, picked, and matched. Consider one popular way we used to sort strings with procedural programming, the "bubble sort." Here is the bubble sort in C++ (link: https://www.geeksforgeeks.org/sorting-strings-using-bubble-sort-2/) :

Code:

```
// C++ implementation
#include<bits/stdc++.h>
using namespace std;
#define MAX 100

void sortStrings(char arr[][MAX], int n)
{
	char temp[MAX];

	// Sorting strings using bubble sort
	for (int j=0; j<n-1; j++)
	{
		for (int i=j+1; i<n; i++)
		{
			if (strcmp(arr[j], arr[i]) > 0)
			{
				strcpy(temp, arr[j]);
				strcpy(arr[j], arr[i]);
				strcpy(arr[i], temp);
			}
		}
	}
}

int main()
{
	char arr[][MAX] =
{"GeeksforGeeks","Quiz","Practice","Gblogs","Coding"};
	int n = sizeof(arr)/sizeof(arr[0]);

	sortStrings(arr, n);

	printf("Strings in sorted order are : ");
	for (int i=0; i<n; i++)
		printf("\n String %d is %s", i+1, arr[i]);
	return 0;
}
```

Result:

```
Strings in sorted order are :
 String 1 is Coding
 String 2 is Gblogs
 String 3 is GeeksforGeeks
 String 4 is Practice
 String 5 is Quiz
```

This code cycles through an array of strings created in main() and called “arr”, which is then sorted in sortStrings(). The array is sorted by going through the array comparing each element with the next and then switching their positions in the array if the next element should come before the first element and repeating this process until all elements are sorted in alphabetical order.

But, to create an intelligence we need to do a lot more with words than sort them in order. To illustrate (we would never program anything this specific in an intelligence), suppose the intelligence needs to remember the name of someone it met. The intelligence knows the name of the person has the letters “rry” in the name, Larry, or something like that, but the intelligence is certain the person’s name is not “Terry” or “Jerry”. In this case we might start out with two lists of names (the code is c#):

Code:

```
var definatelyNot = new List<string> {"Terry","Jerry"};
var nameList = new List<string> {"Abigail","Adah","Adia","Alexandra","Alice",
etc. };

var possibleNames = nameList.Where(p => !definatelyNot.Any(p2 => p2 ==
p)).Where(x => x.EndsWith("rry")).GroupBy(obj => obj).Select(y =>
y.First()).OrderBy(x => x).ToList();
```

Result:

"Barry",
"Harry",
"Larry",
"Perry"

The code declares two lists and then with one line gets the names from nameList that are not in the definatelyNot list and end with “rry”, eliminates any duplicates, and orders the results alphabetically. We can see the abstraction in the declarative statement allows us to do in one line of code something that would be overly complex using a procedural

method. This power of abstraction multiplied by many, many lines of code enables us to create intelligence.

Modern Search Technology

I mentioned sentences earlier, which raises the question: Where does our human-created intelligence get its sentences? There are two options here, the intelligence composes the sentences, or it finds them. With the first option, we need a process for composing sentences. But do I know the process for creating sentences? I do not. I am not conscious of any process for composing a sentence. I just know I need a sentence, become aware of the sentence, and then type it onto the page. Writing code to replicate a process I do not know, or to prove someone else's semantic theory, seems impractical. (Barthes thought a sentence is complete (The Pleasure of the Text, p. 50). Wittgenstein considered the sentence synonymous with the proposition, in which a "a thought finds an expression that can be perceived by the senses (Tractatus Logico-Philosophicus, 3.1).)

It turns out, we have far better tools than ever before for finding a sentence. At one time we could only search for specific strings of text, and augment our searches using "wildcards" such as "%". A search string "car" would not find "cars", so I would use something like "car%", but of course that would return "card", and so our search query gets increasingly complex.

Now search engines such as Lucene use a process call stemming which completes searches by using a word's root. In the above case the search string "car" would find "cars" as well, but not "card".

Search technology offers other features. We can use the "bag of words" approach, which reduces a sentence or sentences, for instance, to a list of words, and we can adjust the results of our search using certain metrics found in the list. We can also search for specific words arranged in a particular way, or certain phrases, and even search using combinations of these searches. Modern search technology also offers something else important. Speed. This is important because our intelligence may need to search for sentences in different ways and choose between the results it gets back. This must happen relatively quickly in a conversation.

Search technology is easy to install on almost any computer. It can be used in many different approaches to creating intelligence.

The Internet

The author and artist Douglas Coupland wrote "I'm 59 and a half years old – and these days I no longer feel that I identify as a human being. I've turned into an app. I'm a filter

for words. I filter the ways I experience the world." Coupland, D. (2021, June 19) Douglas Coupland on Generation X at 30: 'Generational Trashing is eternal.' *The Guardian* https://www.theguardian.com/books/2021/jun/19/ douglas-coupland-on-generation-x-at-30-generational-trashing-is-eternal.

The internet has changed everything about us. The internet has also made human-created intelligence possible. Ri, the intelligence I am working on, has no sensory information and no life history, yet. It must borrow representations of these things, and they are provided by the over 7,000,000 sentences Ri has in its index. I could not provide Ri with 7,000,000 sentences without the internet. The internet also provides Ri with people to talk to, and that's important because intelligence needs social interaction to grow.

The Future

I started this series by defining human-created intelligence as autonomous software that can find its own value, and I believe I have shown it is possible to create such software with tools available to all programmers. These are exciting times. The biggest companies, and countries, have made expensive bets on "AI" and these bets, while they may pay off with limited successes, have failed to get us measurably closer to the dream of an autonomous robot, a dream that many, such as Alan Turing, have had for centuries. Today that dream may be achieved by an individual programmer, or a small group.