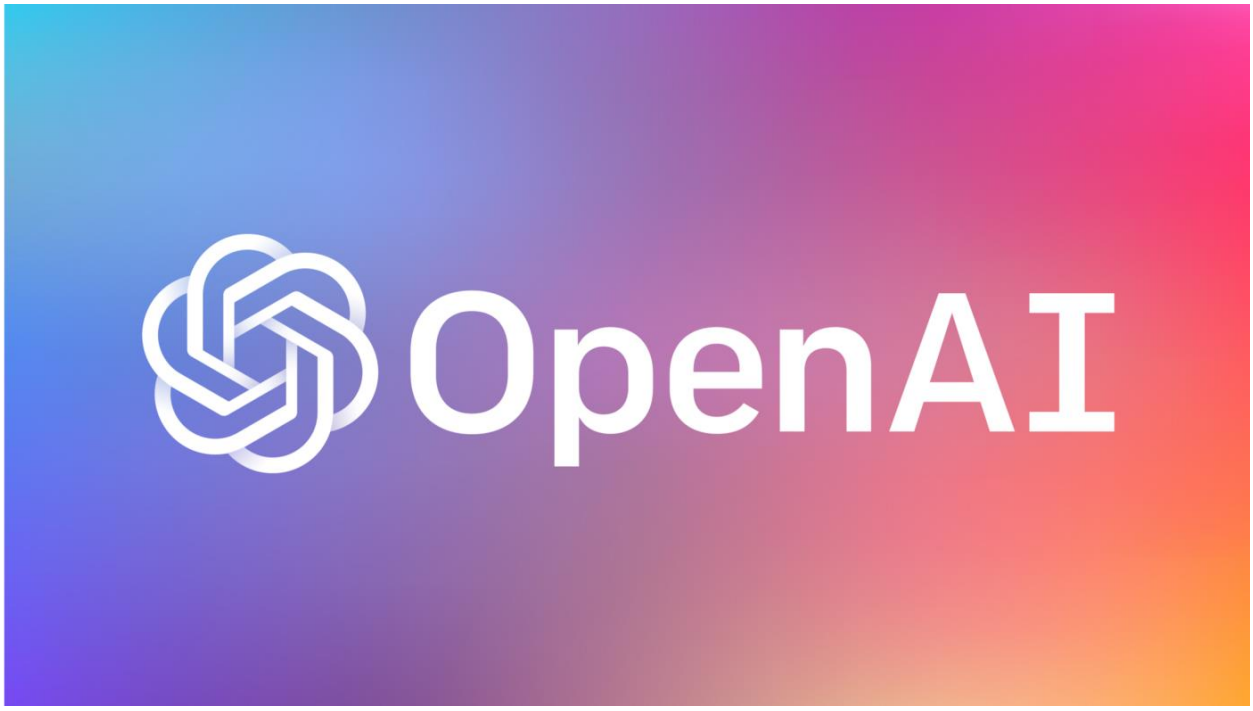

Text to Code

Alvaro Torres Carretero
Daryna Morales Poklad

09.11.2021

Research Goal or Question(s)

With this research, we are trying to see the ability of AI to understand natural language and create code by understanding it.

Methodology

Before starting to code we did some research regarding the different types of AI so we could have a closer idea of the capacity of GPT-3.
Through our investigation, we found tools such as:

- **Demos of GPT-2**, which was able to create code but was not completely useful as it sometimes wrote random code.
- **Codex**, a descendant of GPT-3, is trained to code not only in Python but with Java, PHP, ... However, it still struggles with natural language and there also have been found safety issues.
- **GPT-J** is one of the largest models released to date. Trained on a diverse dataset makes it better than GPT-3. In addition, this tool has image recognition and object-oriented programming. Despite all of that, it is 30 times smaller than GPT-3. However, these are the only skills that are improved regarding GPT-3, a fact that should be considered as it lacks in other areas that GPT-3 does not.

After this research, we started working first with GPT-3 as it is easier to use than GPT-J , to do this we fed it with some SQL examples as it is one of the easiest languages:

- In this test, we will be giving sentences to GPT-3 and the answer that we want to be given to them.
- After giving some examples to the AI, we give a different sentence and GPT-3 gives us the proper answer despite the fact that the sentence is not in the code.
- In conclusion, at the moment we can see that GPT-3 understands natural language on a simple basis even if we change them.

What 's next?

Even though GPT-J looks better at coding, we will continue with GPT-3 because it is easier to work with it. Anyways, our objective now is to try to make GPT-3 able to create a simple database in SQL (via feeding it with some databases and the natural language that explains them) that can lay the foundation for more complex ones.

23.11.2021

Research Goal or Question(s)

This task is focused on how GPT-3 will be trained to create very basic tables in a database giving just the name of the table and if there is any foreign attributes

Methodology

After some tries and fails, we came up with the idea of giving the table name and training GPT3 with a generic type of table.

At first, the idea was not this, it was giving the attributes we wanted to include, but it was not so effective, so we just tried with this model that ended up being so good.

The results we got after just a few examples were great, we tried with three different tables and they came up to be almost perfect, just one of the attributes in the more complex one was not the best, but almost.

Finally, this was our first contact with SQLITE including the tables into a database for later use and better research.

What 's next?

The purpose of this project is to identify how GPT-3 can ease coding tasks by writing simple sentences in natural languages.

As well as in other presentations, once again we can see that training GPT-3 on this task is much simpler than it might seem.

Throughout this process we could see the great performance of GPT-3 with database commands, being a potential help when it comes to the first and basic steps of building and operating with databases.

07.12.2021

Research Goal or Question(s)

This task is focused on how GPT-3 will be trained to add tables given the attributes on the tables and the values.
Similar to the first idea we got for the last sprint but applied for the add statement.

Methodology

As in the first steps, we just added some examples, and after three of them, these statements were easily created by GPT3. This may look superfluous as it is a very basic step but helped us to see how the AI could react to this.

It didn't give us any problems and gave us some time to research a little bit more and to be prepared for the next steps.

What 's next?

For the next step, we will research SQLITE even more, to make sure we understand how it works and try to code some methods that can help us in the future.

21.12.2021

Research Goal or Question(s)

We will go through some examples with SQLITE as it is a relational database management system (RDBMS) contained in a Python library; language which we have been using since the beginning.

Methodology

- **FindNextId:** gives us the following unused id (like auto_increment field in MYSQL)
- **FindIdAttr**: given an attribute and table name, finds its matching id
- **NumberOfAttributes:** shows the total number of attributes added
- **TableColumns**: shows the attributes within the given table name

- **convertTuple, convertToReadable, convertIdToReadable:** transforms given data into readable outputs such as String.

What 's next?

After good research about how SQLITE works, we managed to create some methods that will be useful later in the process.

Next, we will try to use all the data we gathered during all the sprints to use these methods and see how capable of working with those to create databases and sql queries is GPT3.

11.01.2022

Research Goal or Question(s)

This task is focused on how GPT-3 will be trained to use a variable set of methods that will be used to work with databases, e.g.: select, create, modify attributes; connect databases; etc.

Methodology

To test the previous methods, we will need to create and connect a database and tables in it.

01 | Create Connection and tables
02 | Create Database
03 | Create a class named “Example” with which we will work
04 | Use methods created previously

As expected, GPT-3 just needed a few examples to achieve a good level of performance.

Once we have tried Select Statements, new tables will be created and tested with the Add Statement.

Once the results were obtained, we could see that even tables with foreign keys did not have any problem to be created, being this is an important achievement as in

the early stages of the project we were not able to do this.

What 's next?

The purpose of this project is to identify how GPT-3 can ease coding tasks by writing simple sentences in natural languages.

As well as in other presentations, once again we can see that training GPT-3 on this task is much simpler than it might seem.

Throughout this process we could see the great performance of GPT-3 with database commands, being a potential help when it comes to the first and basic steps of building and operating with databases.

For the next and final step, we will be using the fine tuning to create our own model and see how it performs.

25.01.2022

Research Goal or Question(s)

In this task we will go through finetuning, why should we apply it and how.

Why fine tuning? Saving computing time and resources since we avoid most of the training (if we use a pre-trained network and adapt it to our problem) and better results than regular GPT-3.

Methodology

One of the easiest approaches would be to find a default dataset that fits our purpose; it does not have to be our final dataset, as we may change it to make it more suitable for our project. However, we can also build our own dataset from scratch.

In our case, we took data from a dataset called spider that works converting SQL statements into natural language and transformed it for our usage as well as some examples we did by ourselves.

After we obtain our dataset, the data needs to be analyzed and prepared to upload

it to our model.

Once we have trained our model, it will be tested, and the results will be interpreted to analyze the performance of our model.

After accomplishing these steps, the original results will be compared with the fine-tuned ones, expecting the second one to have a better performance than the original model.

1. **Pre-requisites:** last upgrade of *openai package* to be able to use *openai CLI.* This can be done through this command: **!pip install --upgrade openai**
2. **First Step:** as we need to work with GPT-3 models, *export* command will be used to set the needed environment. This can be done through this command: **export OPENAI_API_KEY="<OPENAI_API_KEY>"**

At first, we ran into some problems with these two first steps. They are so basic that it looks like they can't cause errors, but as we were working on Windows, some problems appeared. So, we came up with the idea of using a Linux VM to create an environment where we can work. This caused our fine-tuned model to be created with just 80 examples and our test data is just 25 prompts as the time was against us. Ideally, the model should be fed with hundreds of examples and tested with more prompts than 25.

3. **Second Step:** convert training csv data into a JSONL file. This can be done through this command: **openai tools fine_tunes.prepare_data -f D:\Uni\Stuttgart Uni\Openai\Train_Model.csv**
4. **Third Step:** Once the JSONL has been created we will use it on Davinci as it is the best on code completion. We will be using the next command for fine tuning: **openai api fine_tunes.create -t D:\Uni\Stuttgart Uni\Openai\Train_Model_prepared.jsonl -m davinci**
5. **Fourth Step:** The next command will be used to get our final results for our fine-tuned model: **openai api completions.create -m davinci:ft-hdm-university-stuttgart-2022-01-24-20-12-41 -p "Translate this prompt to SQL: what percent of users who signed up converted to paid?->"**
6. **Fifth step:** We will repeat the same command but using davinci instead of our model: **openai api completions.create -m davinci -p "Translate this prompt to SQL: what percent of users who signed up converted to paid?->"**

7. **Final Step:** And finally, we will use the Python code we coded in the first sprints with a few examples to compare this to these results and have a better perspective on how it evolved.

Conclusion

Results will be gathered in a xlxs file where percentages of correct results will be displayed. As expected, the fine-tuned model achieves better results than the original one.

Also, unexpectedly, the base model of davinci performed better than the model with a few examples of the first sprints. This could be because of the complexity of the statements we tried, these statements were, in their majority, more complicated than the ones we used to feed GPT3.