Name of student : TARCIN Audrey

Name of your Level 1: Shai NAKACHE

Title : Automated Source Code Generation and Auto-Completion Using Deep Learning: Comparing and Discussing Current Language Model-Related Approaches

Keywords specific to the paper: Deep learning for code generation, NLP and code dataset, Python code generation, Deep Neural Networks

The first three pages deal with the use of deep learning in language models to automatically generate code sources. It refers to different deep learning architectures to create language models based on programming code and compares neural network architectures like AWD-LSTM, QRNN and Transformers that uses transfer learning techniques on a Python dataset to generate code. The other part looks at the methods used in the research as well as the selection of architectures mentioned above and also tokenization models such as BPE and Word Piece methods.
They then performed an experiment to train and evaluate their code generation and auto-completion task performance on a python dataset from GitHub CodeSearchNet Challenge.)

The document also discusses the dataset used in natural language processing (NLP) research and source code.
Specific details include using Python to compare with the existing literature, the software used (FastAI, Google SentencePièce, Hugging Face's Transformers) and the pre-processing steps of the dataset.
It highlights the use of models such as GPT-2, BERT and RoBERTa, training procedures, transfer learning, tokenization methods and hardware specifications. Analysis of the results of the AWD-LSTM, AWD-QRNN and Transformer models, presenting precision measurements, validations, and comparison with previous models in terms of code generator. Examples of source code generator and auto-completion from trained models are provided for the evaluation of the quality of model results. The section also specifies the importance of human evaluation in addition to measures in the evaluation of model performance.)

This part deals with the evaluation of the python code generator using other language models (GPT-2, AWD-LSTM, AWD-QRNN, BERT, and RoBERTa). He says that GPT2 gives better results in terms of accuracy than python. That of BERT and RoBERTa shows less accurate results. The paper mentions the need for future research to further explore the impact of different training methods in the field of source code generator. It also highlights the importance of evaluating the code generated by the human and allows to create new measures for evaluating tasks

The doc deals with the comparison of deep neural network architectures differences
It highlights that small models work better, while those like gpt2 work well with a source code generator that uses different approaches to tokenization. Models like BERT and ROBERTA

are accurate but have difficulties for specific tasks. Pre-trained models have better performance.
The document highlights gaps in the automation of source code generator and auto-completion tasks.

AI Model used:

AWD-LSTM: Attention-based Long Short-Term Memory (LTM)
AWD-QRNN: Attention-based Quantized Recurrent Neural Network (AWD-QRNN)
Transformers: Attention-based neural network architecture (Attention-based neural network architecture)
GPT-2: Generative Pre-formed Transformer (Generative Pre-trained Transformer) Model
BERT: Bidirectional Transformer Model for Language Representation (Bidirectional Encoder Representations from Transformers)
RoBERTa: Robust version of BERT (Robustly optimized BERT pretraining approach)
BPE (Byte Pair Encoding): Divides words into frequent character pairs
WordPiece: Divides words into morphological subunits
FastAI: Open-source library for machine learning
Google SentencePiece: Tokenization tool
Hugging Face’s Transformers: Open-source library for transformer models