MUNI
FI

AI-driven Software Development
Source Code Quality

Petr Kantek

Introduction

- First segment
  - Bit of theory behind code generation models
  - Natural language processing (NLP) tasks
  - Transformers
  - Large Language Models (LLMs)
- Second segment
  - Problem statement
  - Source code quality
  - Tools for code generation
  - Sample of experiments

First Segment

MUNI FI

Naturalness of Software 1/2

- We are able to create statistical language models of natural language
- In paper „On the naturalness of software“ from 2012, authors show that software contains similar patterns as natural language
  - Although potentionally complex, only small fraction is used
  - A lot of repetitions
  - Clear patterns that can be statistically modelled
  - Cross-entropy of source code corpus is smaller than of English corpus
- And thus, code generation could be handled as a standard Natural Language Processing (NLP) task

Naturalness of Software 2/2

- Using N-Gram language model, authors implemented an Eclipse plugin for code completion
- N-Gram = given local context (previous tokens) I can get the most likely next token
- For smaller token sequences worked relatively well
- Foundation of using standard NLP tools for code generation, instead of language-based rules (for instance type context)

NLP Tasks for Source Code 1/5

Task Category	Task
Code-Code	Clone Detection Defect Detection Code Completion Code Suggestion Code Repair Code Translation
Text-Code	NL Code Search Text-to-Code Generation
Code-Text	Code Summarization
Text-Text	Documentation Translation

MUNI FI

NLP Tasks for Source Code 2/5

- Code Generation
  - Code generation is an an automated process of transforming natural language specifications or descriptions into executable source code
  - Natural language specifications can be in the form of code-level comments, prompts, documentation and other
- Code Completion
  - A feature that suggests and automates the insertion of code elements as developers type
  - A common feature of integrated development environments (IDEs)
- Code Suggestion
  - Subtask of code generation providing developers with intelligent recommendations of code snippets for code enhancements, optimizations, or alternative implementations

NLP Tasks for Source Code 3/5

- Code Translation
  - Also called transpilation
  - Code translation is the conversion of code from one programming language to equivalent code in another programming language
  - Enables interoperability and adaptation across diverse programming languages, technological environments, and domains
  - Transpiler
- Code Refinement
  - Improving or optimizing existing source code
- Code Summarization
  - Creating concise and informative summaries of code snippets or entire codebases to facilitate comprehension, documentation, and knowledge transfer
  - Useful for legacy codebases

NLP Tasks for Source Code 4/5

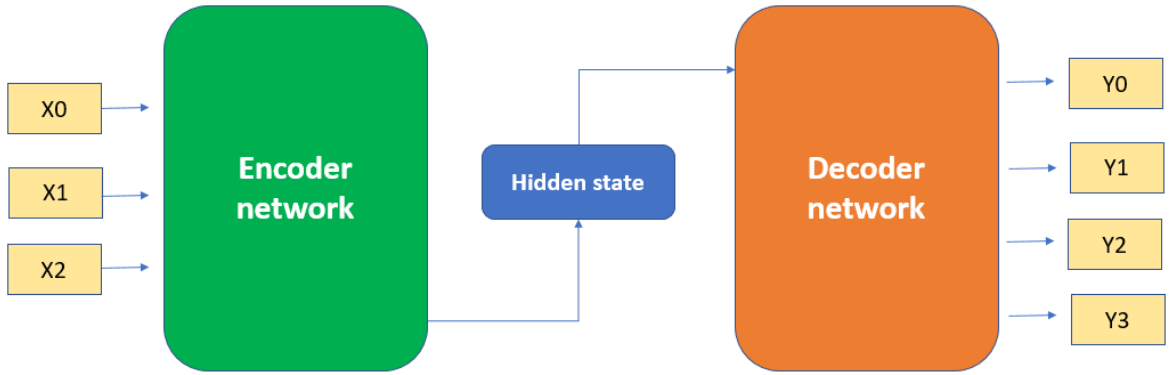
- Defect Detection
  - The identification and analysis of bugs, errors, or imperfections in software code to improve its correctness, reliability, and functionality
  - Can be implemented as a binary classification task, where the input code snippet is categorized either as defective or correct
- Code Repair
  - Automatic or semi-automatic techniques for identifying and fixing issues or errors in source code
  - Additional functionality on top of defect detection
  - Self-healing applications rewrite themselves by prompting AI
- Clone Detection
  - Identifying redundant or similar sections of code within a software project
  - DRY principle

NLP Tasks for Source Code 5/5

- Documentation Translation
  - The translation of software documentation from one language to another
  - Close to common NLP tasks such as machine translation
- NL Code Search
  - Search for relevant code snippets using natural language „queries“
  - Contextual descriptions rather than full-text search

Transformer Neural Architecture 1/2

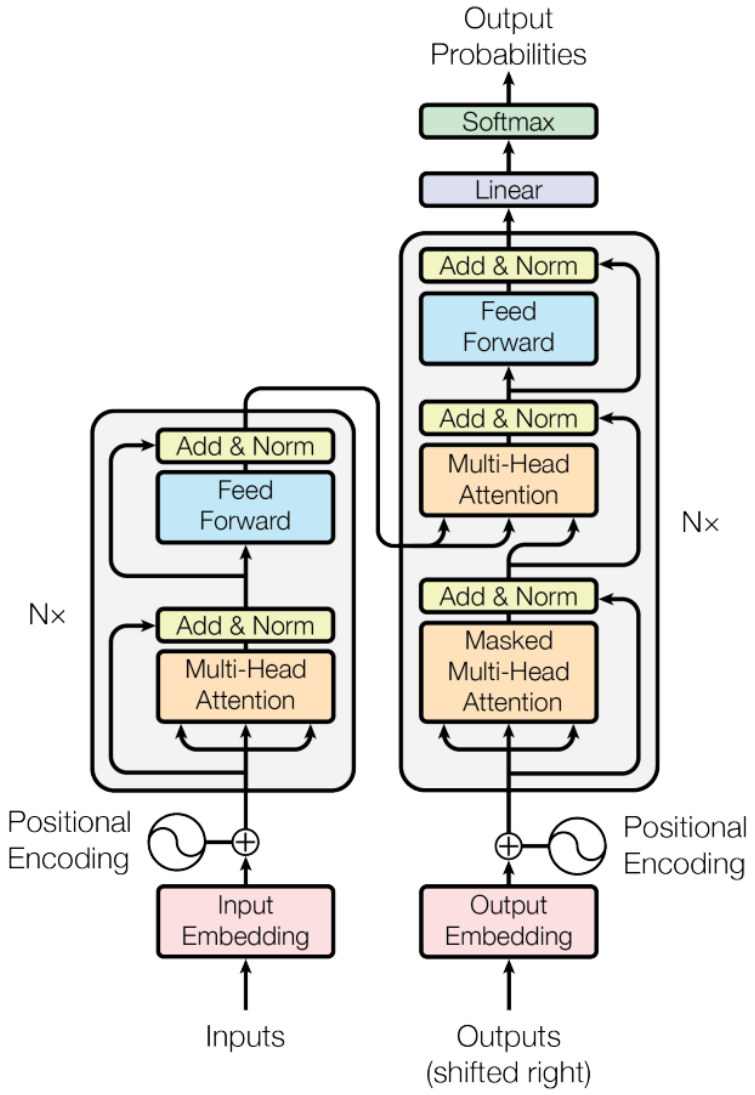
- Encoder
  - Generates
    - Hidden state
    - Embeddings of semantic/syntactic information
- Decoder
  - Generates output
- Encoder & Decoder
  - Machine translation
  - Natural language description to code
- Not just for text
  - Vision Transformers



MUNI FI

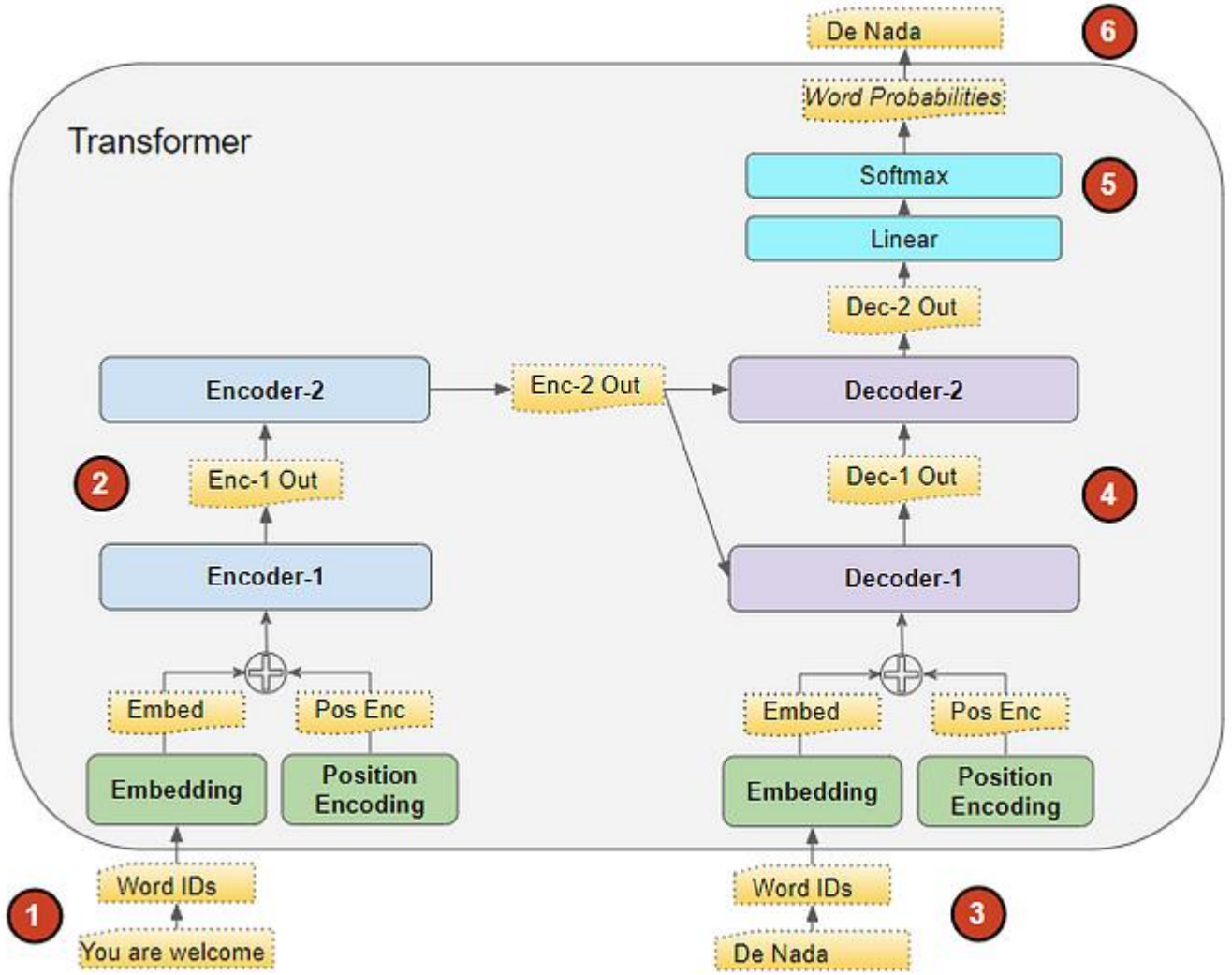
Transformer Neural Architecture 2/2

- Tokenization of input
- Input embedding
- Positional encoding
- Multi-Head attention
- Feed forward network
- Layer normalization
- Softmax for output

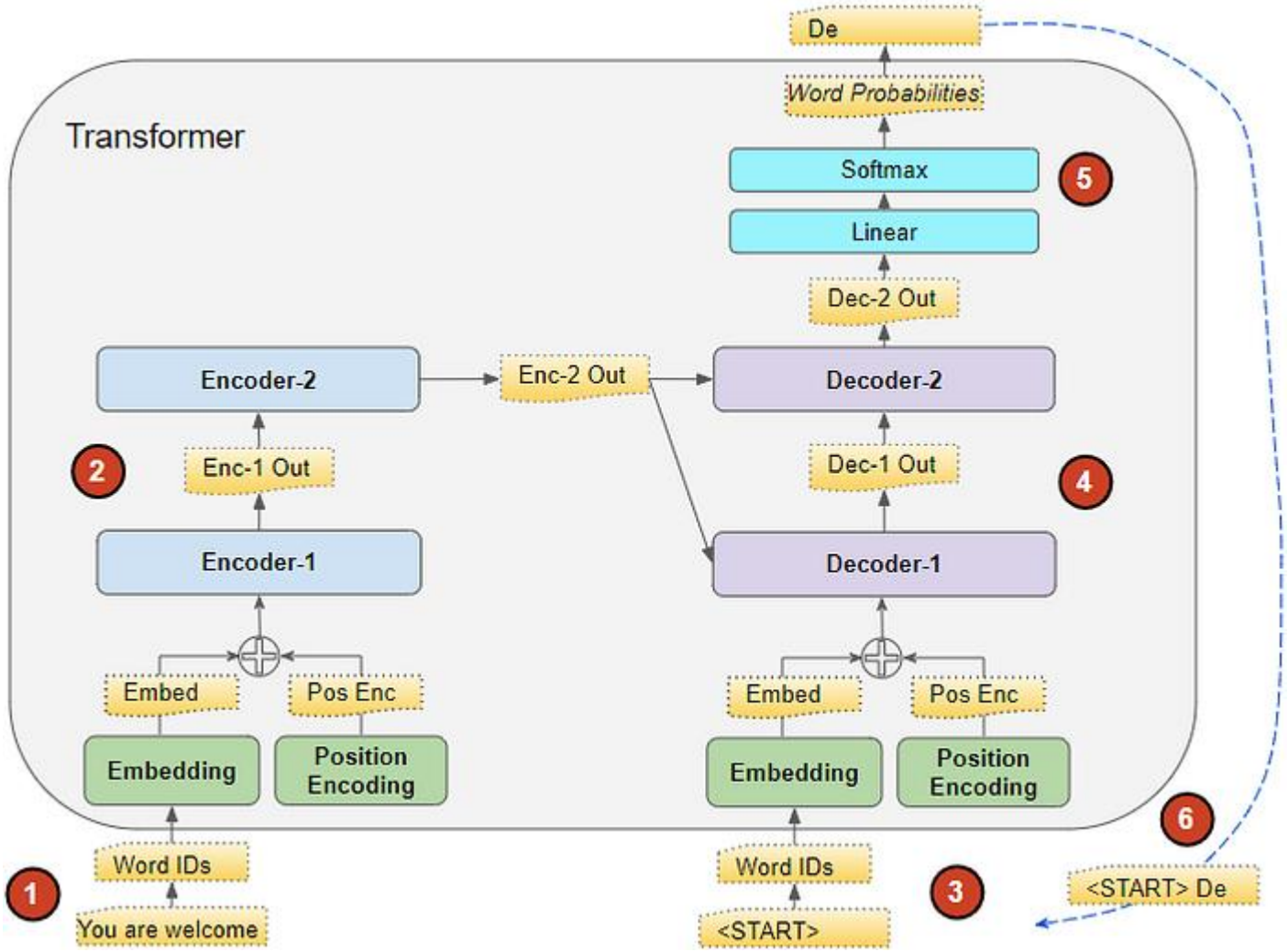


MUNI FI

Transformers Training



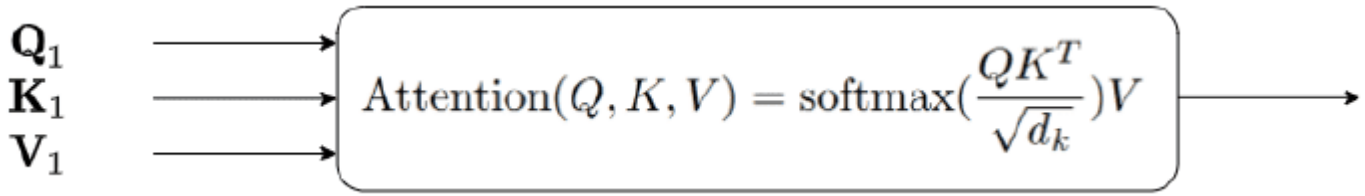
Transformers Inference

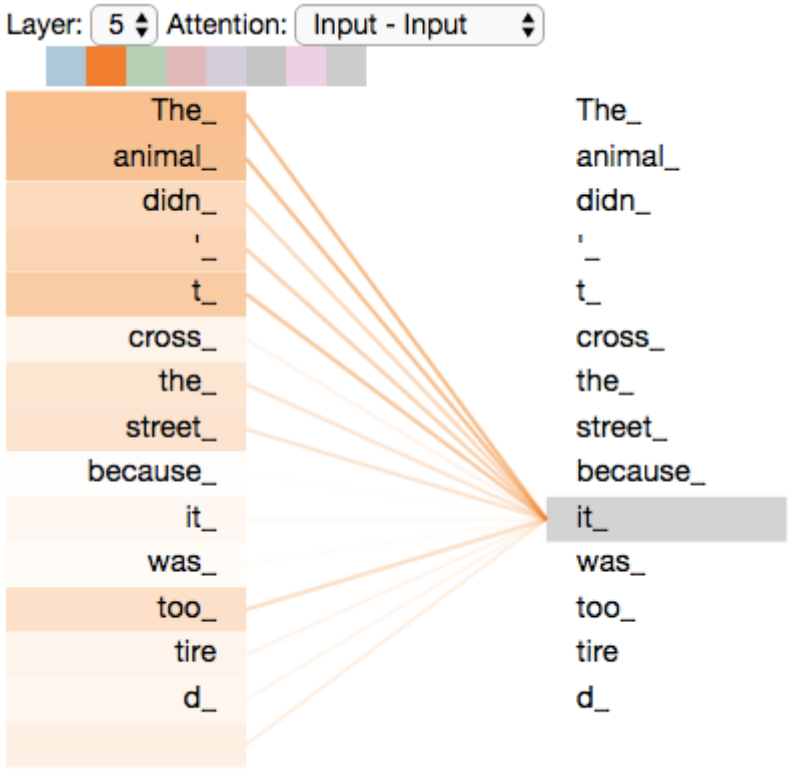


MUNI FI

Attention

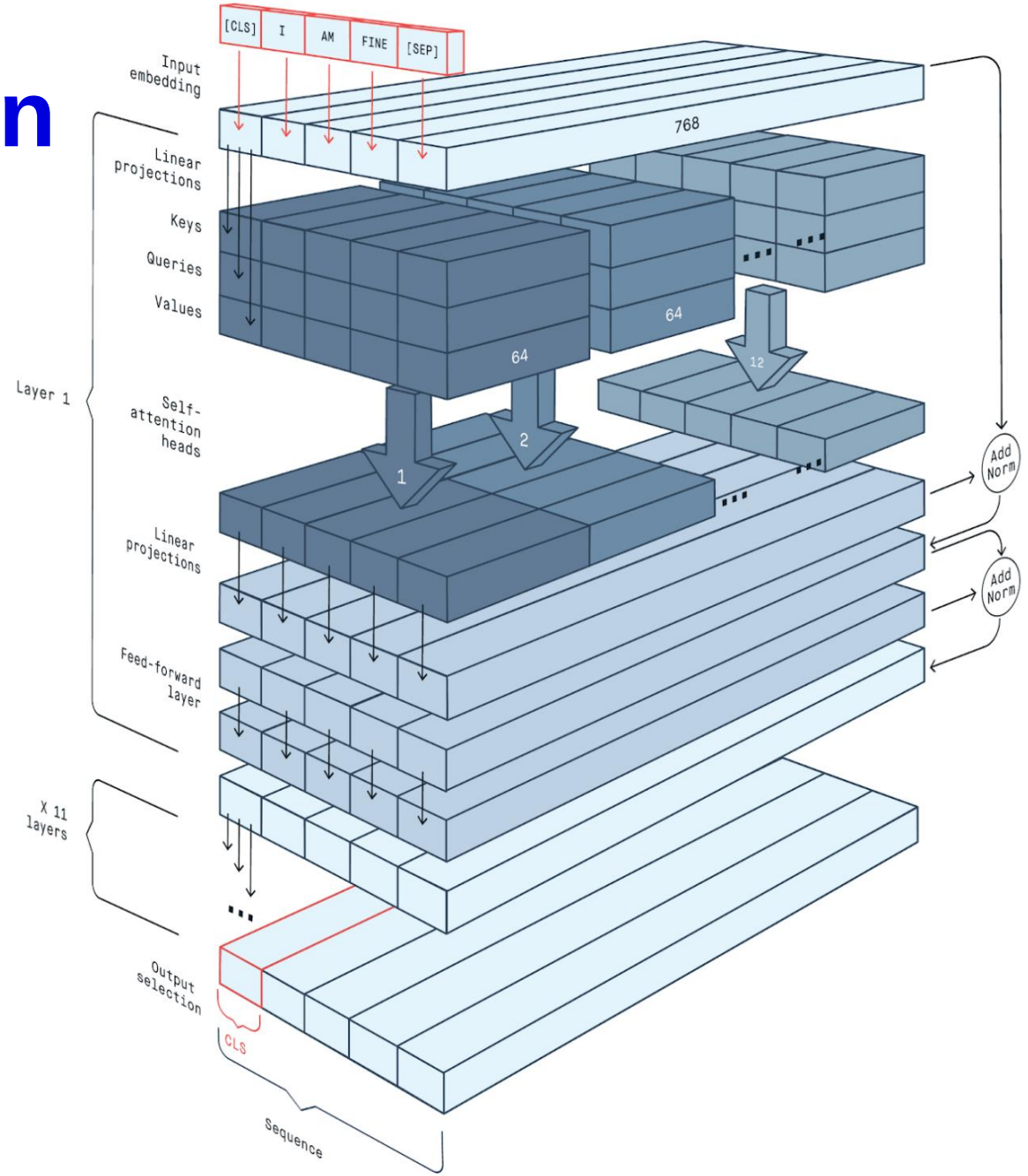
- Function
  - Resembling retrieval of information
  - Query
    - What I am searching for
  - Keys
    - Description of the information available
  - Values
    - The actual information
- Self-attention
- Scaled dot product attention
- Multi-head attention





MUNI FI

Multi-head Attention

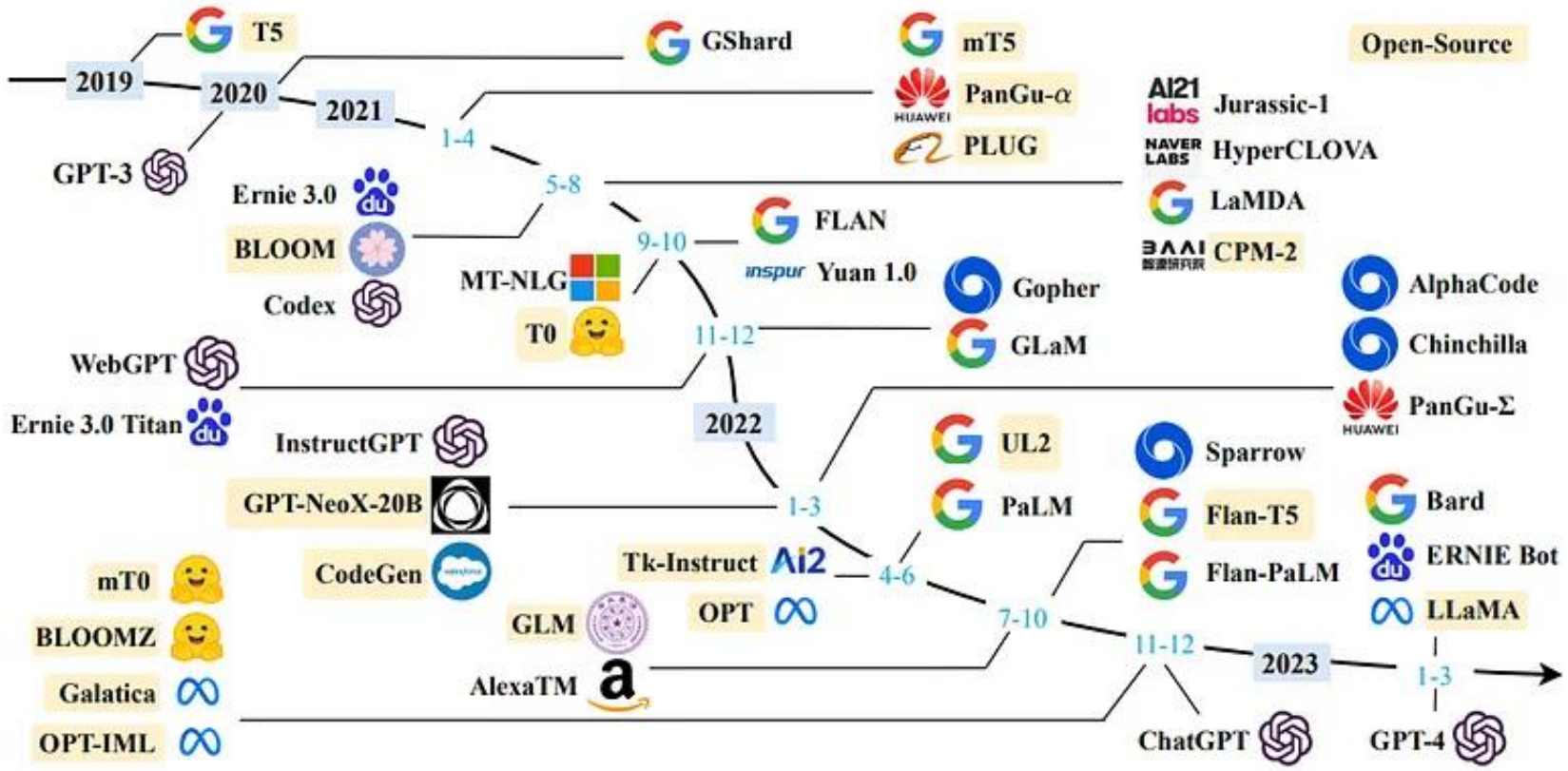


MUNI FI

Attention Rollout

- https://alphacode.deepmind.com

Large Language Models (LLMs)



MUNI FI

Large Language Models (LLMs)

- Generative models
- Based on Transformer architecture
  - Mostly using only decoder part
- Prompting
  - Chatbots
  - Giving instructions to LLMs
- Either general LLM or fine-tuned to downstream NLP tasks
  - Source code NLP tasks
- Self-supervised pre-training
  - On vast amounts of text
  - Generate the next word in the sentence

LLMs Fine-tuning

- Instruction-based tuning
  - The model is provided with user‘s message and generates prediction/answer
  - It then tries to minize the difference between predictions and correct answers
- Reinforcement Learning from Human Feedback
  - To maximize helpfulness
  - Minimize harm
  - Avoid dangerous topics
  - Based on Reinforcement learning – environment, rewards
  - Model generates multiple predictions and human ranks them from best to worst
  - Aligns predictions with human preferences

Large Language Models (LLMs)

- OpenAI: GPT-1, GPT-2, GPT-3, GPT-3.5, GPT-4, GPT-5, Codex
  - GPT-1/2 open-source
  - The rest closed-source
- Meta: Llama 1, Llama 2, Llama Code
  - Open-source
- HuggingFace: Falcon, CodeParrot
  - Open-source
- DeepMind: Chinchilla, AlphaCode
  - Cloused-source

Risks of LLMs

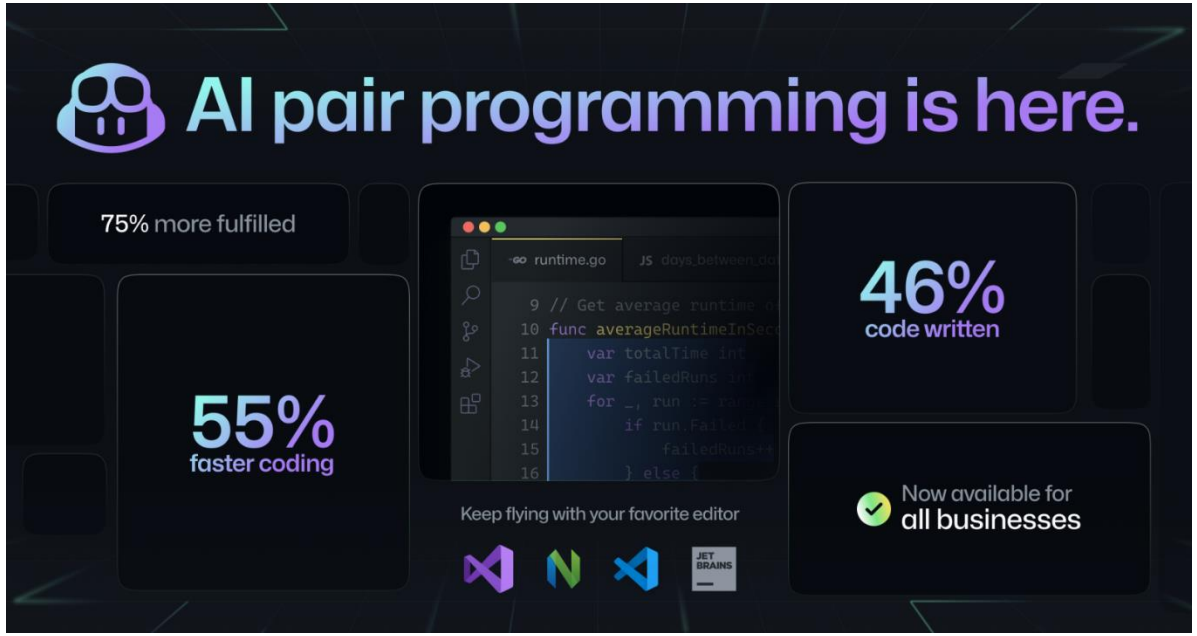
- Known for not being optimal models in terms of risks and security
- Mostly due to the fact that LLMs are trained on diverse datasets harvested from public internet
- Bias
  - propensity to favor or disfavor particular groups or concepts based on the patterns observed in the training data
- Hallucinations
  - Refer to instances where the model generates content that is factually incorrect, fictional, or entirely fabricated
  - Code snippets

Risks of LLMs

- Trustworthiness
  - Reliability and dependability of the information generated
- Racism
  - generation of content that discriminates against or perpetuates stereotypes about individuals or groups based on their race or ethnicity
- Security
  - potential vulnerabilities, that could be exploited to manipulate or compromise the model or its output
  - Indirect vulnerabilities by freely using whatever code LLMs generate
- Toxicity & Hate Speech
  - generation of content, that is harmful, offensive, or abusive

Second Segment

Problem Statement



- Code generation models can output low quality code
  - Can contain vulnerabilities
  - Type errrors
  - Non-existing libraries or syntax
  - Might break best practices principles
  - Or might not work at all
- Experiment with various LLM-based code generation tools

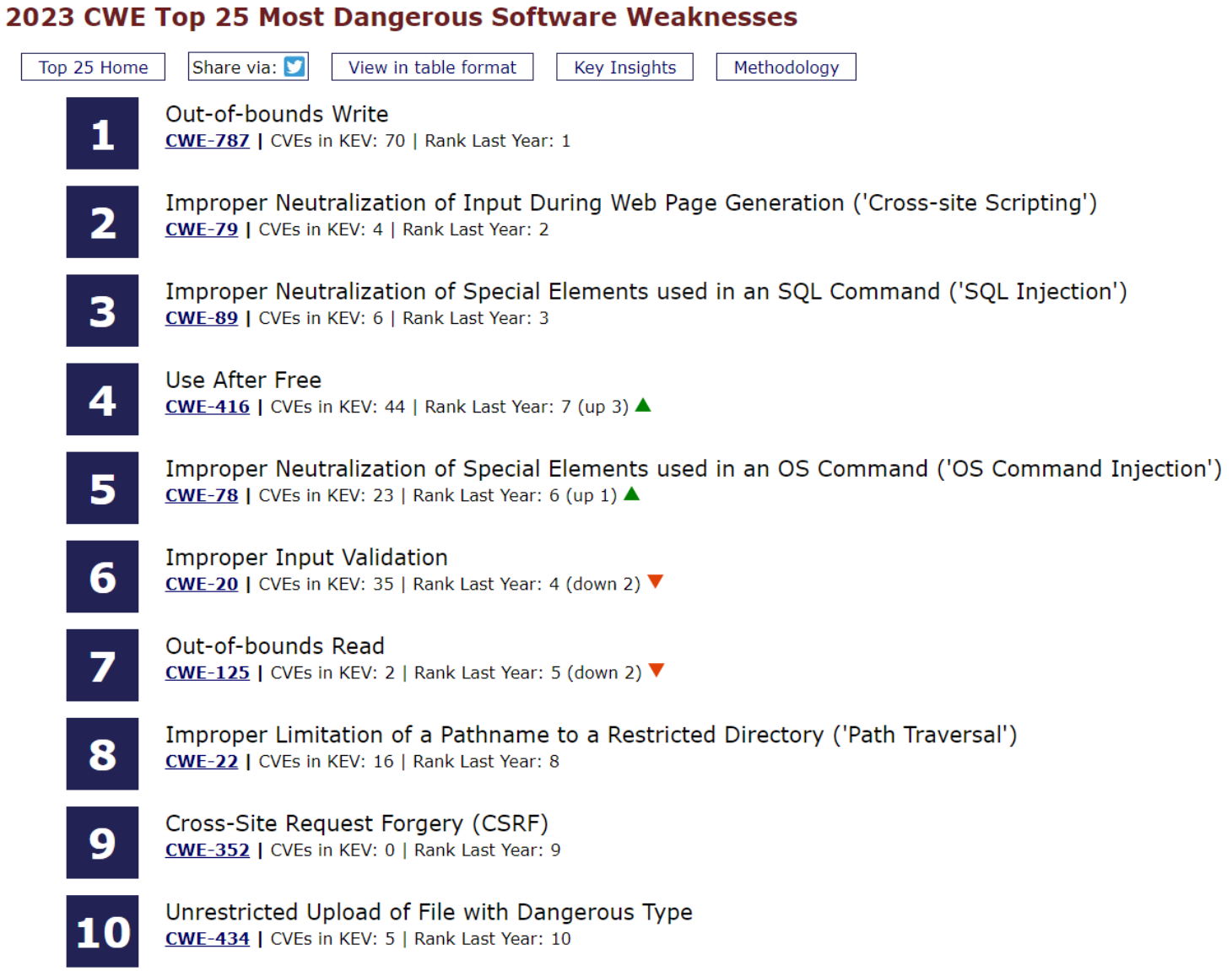
MUNI FI

Software Vulnerabilities

- Software vulnerabilities are weaknesses or flaws in software code that can be exploited by attackers to compromise the security or functionality of a system
- SQL Injection
- Cross-site Scripting
- Authorization Attacks

Common Weakness Enumeration

- A list of most common weaknesses/vulnerabilities
- Every year a top 25 most dangerous weaknesses
- Hierarchical structure



Source Code Quality

- Static analysis
- Linters
  - Check source code against a set of rules
  - Adherence to style guides
  - Python linter Mypy checking against PEP8
- Source code metrics
  - Lines of code
  - Average number of methods
  - Cyclomatic complexity
  - …

CodeQL

- Static analyzer from GitHub
- Specific query language
- Queries for CWE detection
- CLI or CI/CD

UnsafeDeserialization.ql

```
import TaintTracking::Global<UnsafeDeserializationConfig>

from PathNode source, PathNode sink

where flowPath(source, sink)

select sink.getNode().(UnsafeDeserializationSink).getMethodAccess(), source, sink,
  "Unsafe deserialization of $@.", source.getNode(), "user input"
```

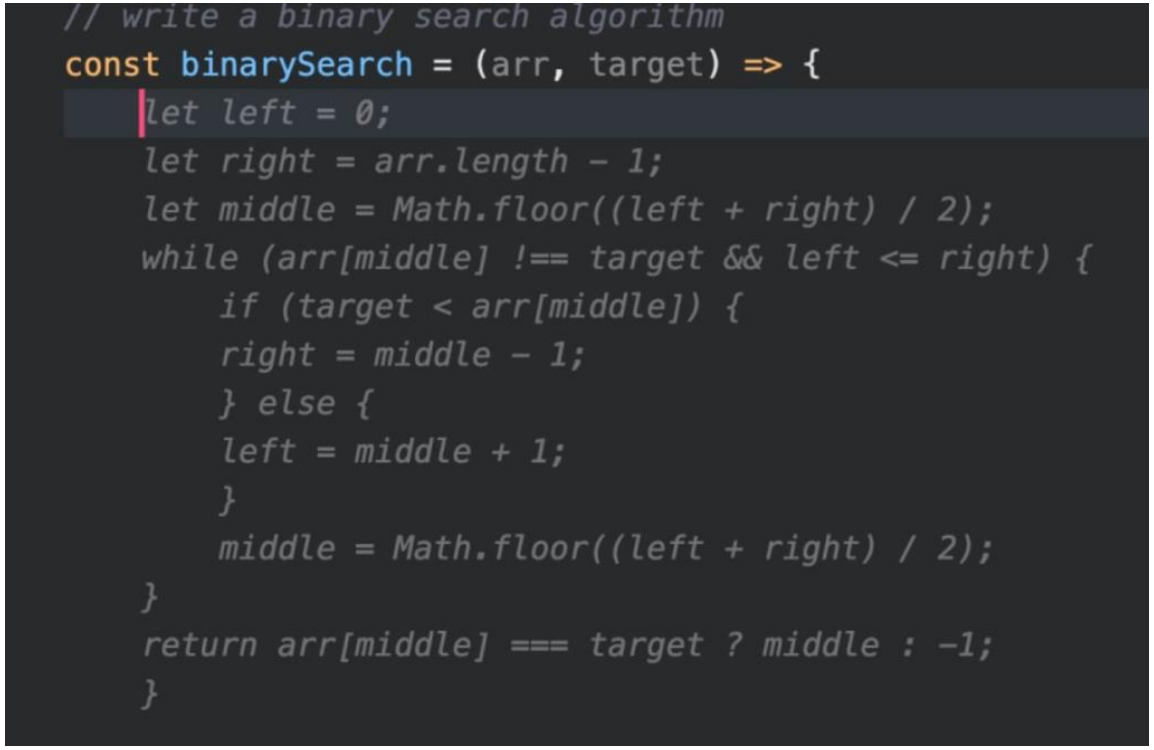
Overview

CWE	Language	Query id	Query name
CWE-11	C#	cs/web/debug-binary	Creating an ASP.NET debug binary may reveal sensitive information
CWE-12	C#	cs/web/missing-global-error-handler	Missing global error handler
CWE-13	C#	cs/password-in-configuration	Password in configuration file
CWE-14	C/C++	cpp/memset-may-be-deleted	Call to `memset` may be deleted
CWE-20	C/C++	cpp/count-untrusted-data-external-api	Frequency counts for external APIs that are used with untrusted data
CWE-20	C/C++	cpp/count-untrusted-data-external-api-ir	Frequency counts for external APIs that are used with untrusted data
CWE-20	C/C++	cpp/untrusted-data-to-external-api-ir	Untrusted data passed to external API
CWE-20	C/C++	cpp/untrusted-data-to-external-api	Untrusted data passed to external API
CWE-20	C/C++	cpp/uncontrolled-process-operation	Uncontrolled process operation
CWE-20	C/C++	cpp/unclear-array-index-validation	Unclear validation of array index
CWE-20	C/C++	cpp/uncontrolled-allocation-size	Overflow in uncontrolled allocation size
CWE-20	C/C++	cpp/late-check-of-function-argument	Late Check Of Function Argument
CWE-20	C/C++	cpp/linux-kernel-no-check-before-unsafe-put-user	Linux kernel no check before unsafe_put_user vulnerability detection
CWE-20	C#	cs/count-untrusted-data-external-api	Frequency counts for external APIs that are used with untrusted data

MUNI FI

AI Tools for Code Generation

- GitHub Copilot
  - Built on OpenAI Codex model
  - VS Code extension
  - Paid license
- TabNine
  - Combination of GPT models
  - No additional information about architecture
  - VS Code extension
  - Free & Paid license
- ChatGPT
  - Web interface
  - 3rd Party VS Code plugins need OpenAI API key



```
// write a binary search algorithm
const binarySearch = (arr, target) => {
    let left = 0;
    let right = arr.length - 1;
    let middle = Math.floor((left + right) / 2);
    while (arr[middle] !== target && left <= right) {
        if (target < arr[middle]) {
        right = middle - 1;
        } else {
        left = middle + 1;
        }
        middle = Math.floor((left + right) / 2);
    }
    return arr[middle] === target ? middle : -1;
    }
```


MUNI FI

Experiments – RQ1

- Which tool from GitHub Copilot, TabNine, and ChatGPT is able to suggest the least vulnerable Python code according to a defined list of 5 CWEs (sql injection, ssh missing host key, server-side cross-site scripting, …)
- 25 code snippets x 3 lengths (short, medium, longer) per tool

Tool	Number of snippets	# Containing vulnerabilities	% Containing vulnerabilities
GH Copilot	75	12	0.16
TabNine	75	17	0.22
ChatGPT	75	21	0.28

Experiments - RQ2

- Which tool from GitHub Copilot, TabNine, and ChatGPT is able to suggest code with least amount of Python linting errors
- Python linter Mypy in strict mode
- 25 code snippets x 3 lengths (short, medium, longer) per tool

Tool	Number of snippets	# Containing errors	% Containing errors	# Total errors
GH Copilot	75	43	0.57	155
TabNine	75	42	0.56	172
ChatGPT	75	40	0.53	168

Experiments – RQ3

- Which tool from GitHub Copilot, TabNine, and ChatGPT is able to write the most adhering Python docstring to PEP8 given function signatures
- Python linter Pydocstyle
- 25 code snippets x 3 lengths (short, medium, longer) per tool

Tool	Number of snippets	# Containing errors	% Containing errors	# Total errors
GH Copilot	75	3	0.04	8
TabNine	75	5	0.06	6
ChatGPT	75	12	0.16	27

Sources

- A. Hindle, E. T. Barr, Z. Su, M. Gabel and P. Devanbu, "On the naturalness of software," 2012 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, 2012, pp. 837-847, doi: 10.1109/ICSE.2012.6227135.
- Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
- https://cwe.mitre.org
- https://github.com, https://www.tabnine.com, https://chat.openai.com

MUNI FI

Thank You

MUNI
FI