Automated Software Vulnerability Detection with Deep Learning for Natural Language Processing

Noah Ziems (*Research Scientist*)
Shaoen Wu (*Endowed Chair Professor*)

School of Information Technology
Illinois State University

CENTER FOR CYBERSECURITY
RESEARCH AND EDUCATION
Illinois State University

Literature Background

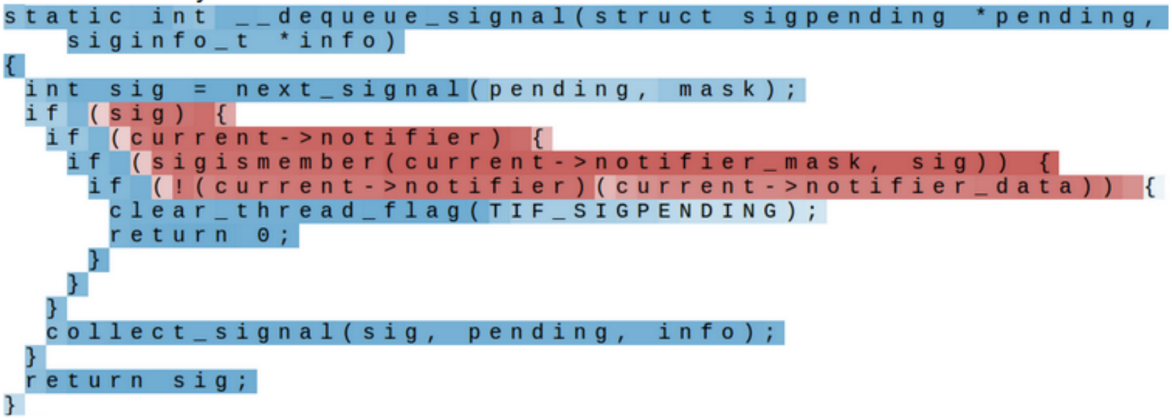
Traditional Methods for Detecting Security Vulnerabilities

- Static Analysis
  - Pre-written set of rules
  - Does not execute code
  - **Prone to false positives**
- Dynamic Analysis
  - Unit Tests
  - Written by programmer
  - Executes code
  - **Vulnerabilities must be anticipated**

Deep Learning Natural Language Processing for Code

- Very good at finding patterns in text
- Previous Work on Code
  - Karpathy et al.(2015)
  - Lachaux et al.(2020)
  - Rozerie et al.(2021)
  - OpenAI Codex(2021)

Cell that robustly activates inside if statements:

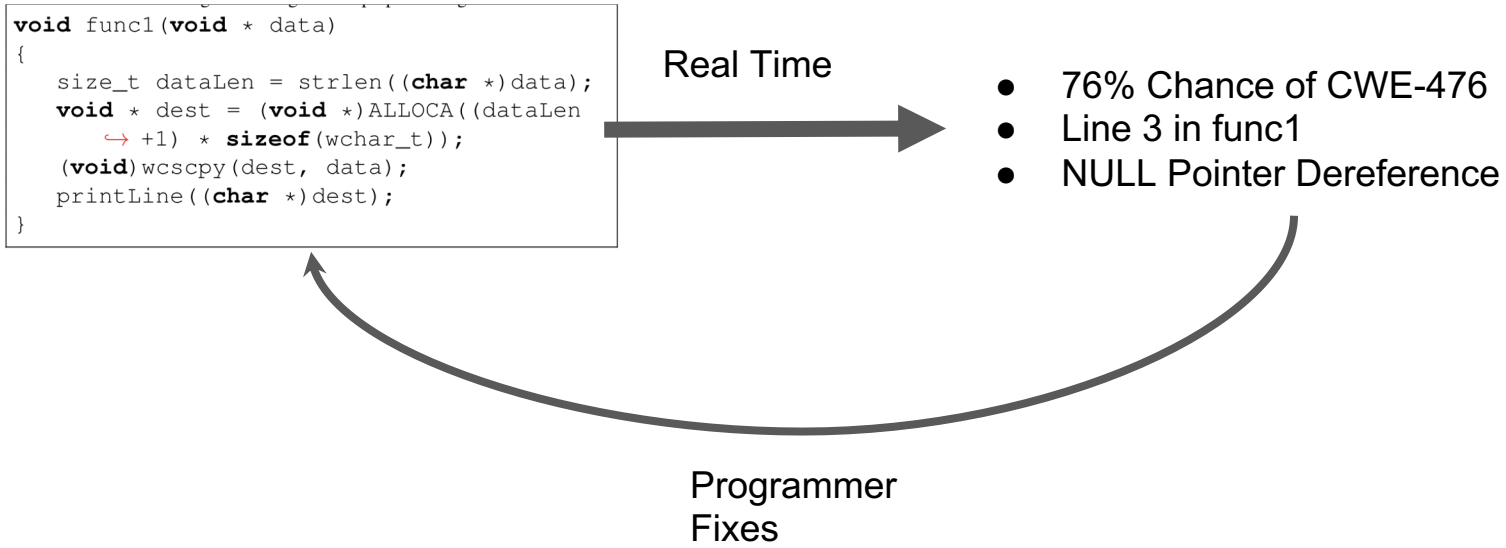


Software Vulnerability Detection with DL-NLP

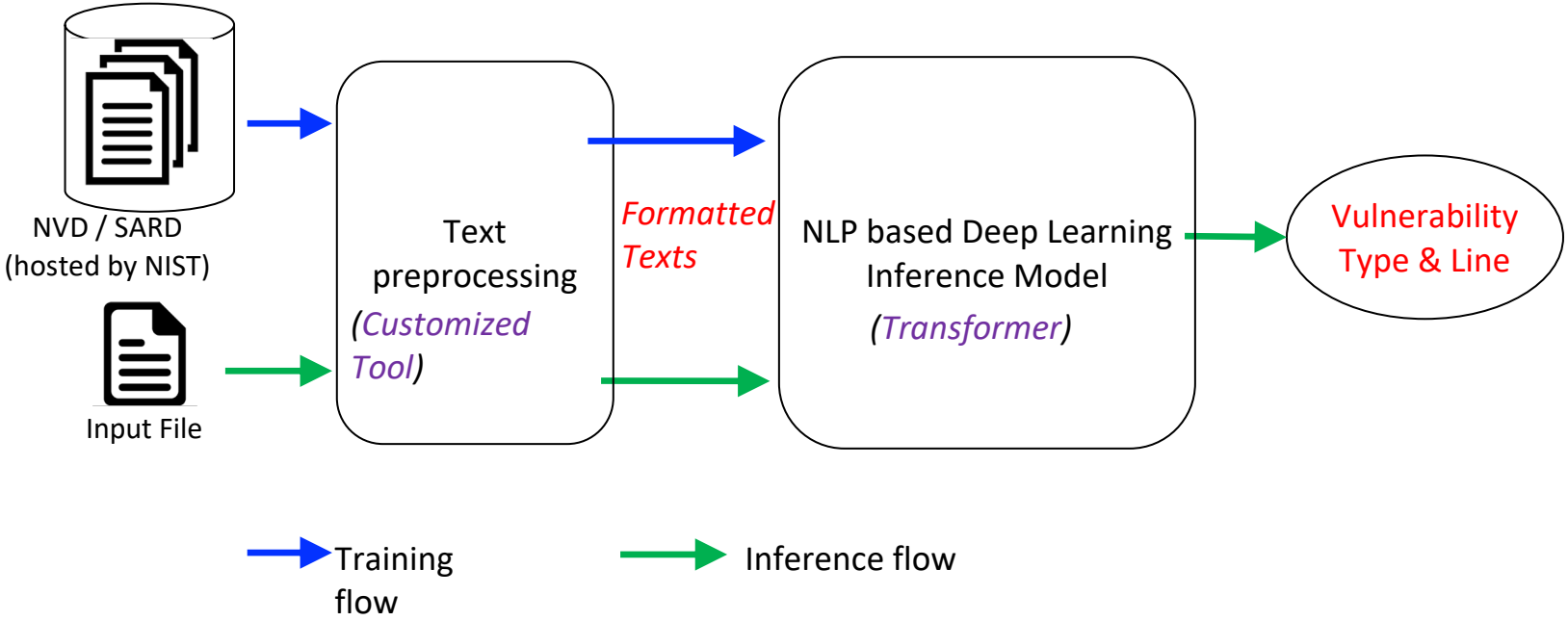
Objectives

- General
  - Warn coders of security vulnerabilities in C++/Java
  - Specific Line
- Technical Details
  - State of the Art Transformer Language Model(DOBF)
  - Pretraining on C++/Java
    - Open Source Github Projects
  - Custom tokenizer using clang python library(C++)
  - function-wise evaluation

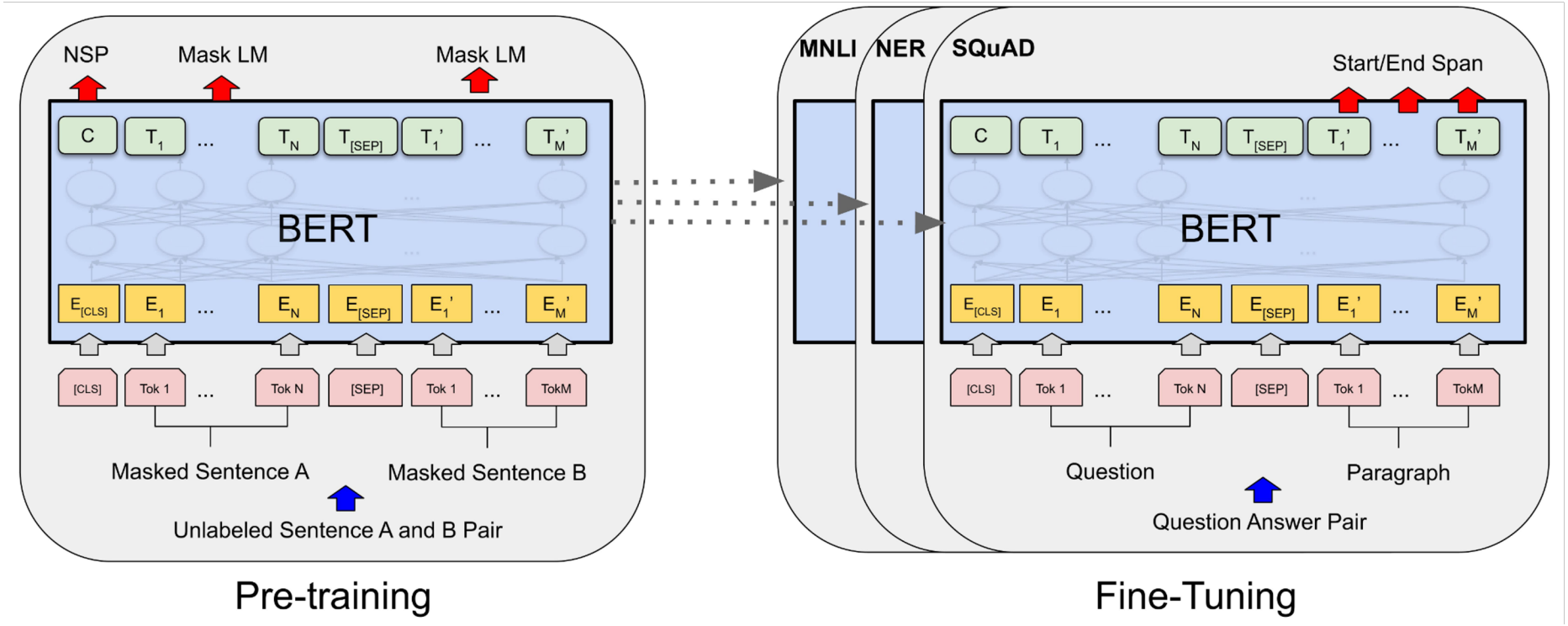
In Summary


void func1(void * data)
{
size_t dataLen = strlen((char *)data);
void * dest = (void *)ALLOCA((dataLen
↪ +1) * sizeof(wchar_t));
(void)wcscpy(dest, data);
printLine((char *)dest);
}
Real Time
● 76% Chance of CWE-476
● Line 3 in func1
● NULL Pointer Dereference
Programmer
Fixes

Dataset

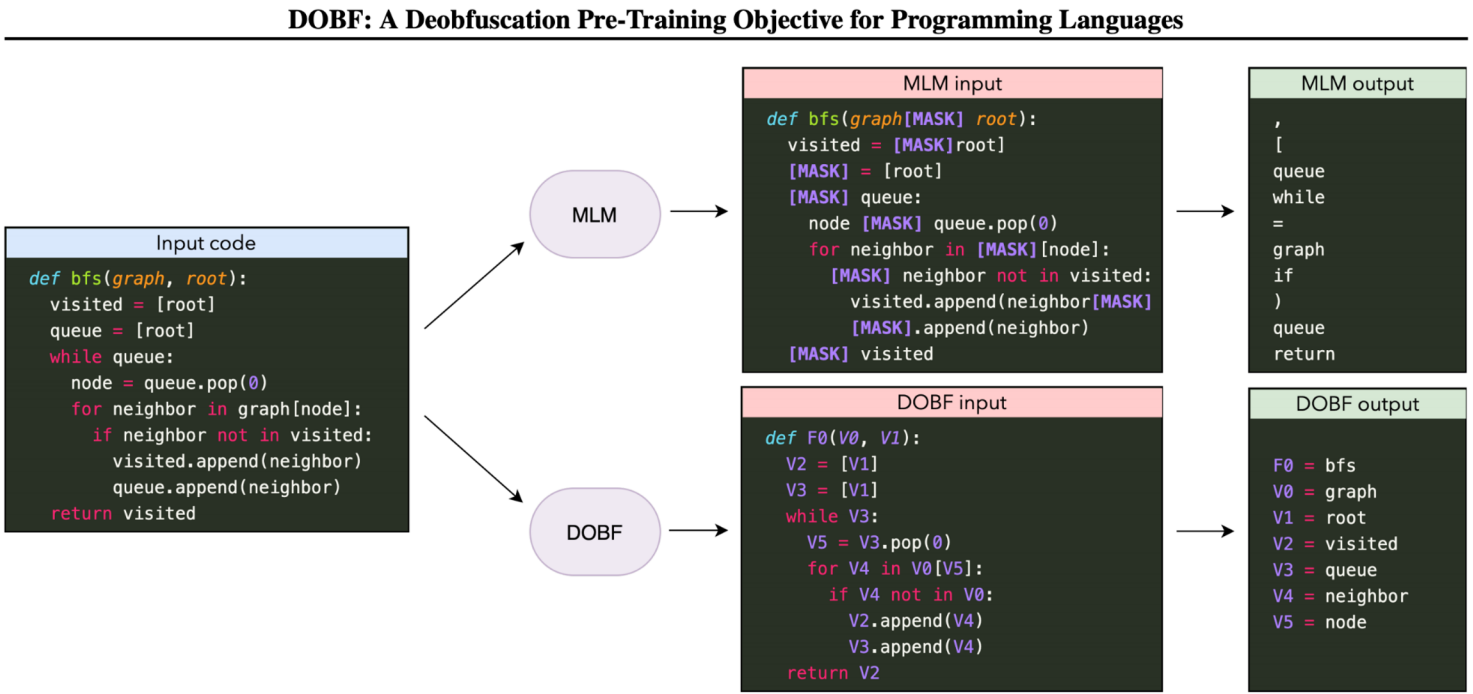

NVD / SARD
(hosted by NIST)
Input File
Text
preprocessing
(Customized
Tool)
Formatted
Texts
NLP based Deep Learning
Inference Model
(Transformer)
Vulnerability
Type & Line
Training
flow
Inference flow

Pre-Training Intuition



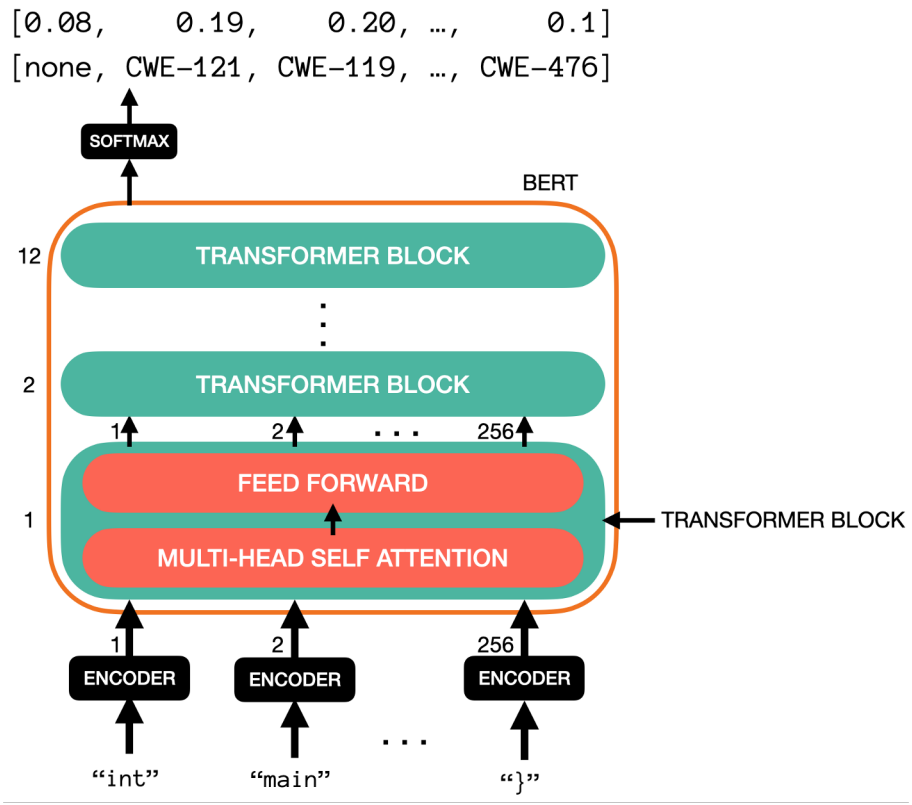
Source: Devlin et. al (2019)

Pre-Training & Custom Tokenization



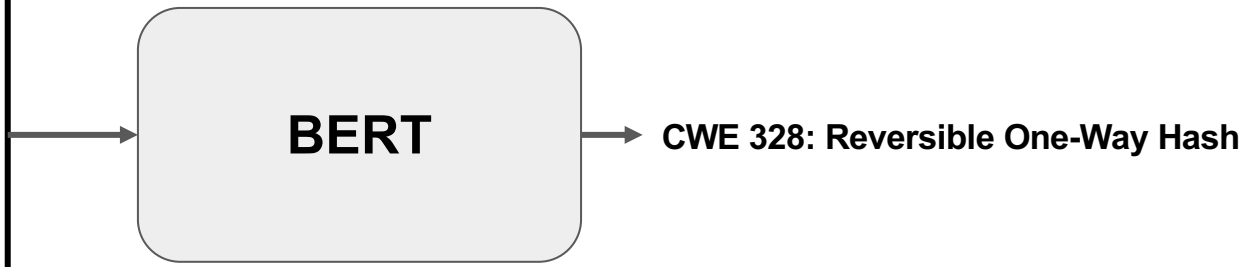
Source: Roziere et. al (2021)

Architecture Diagram



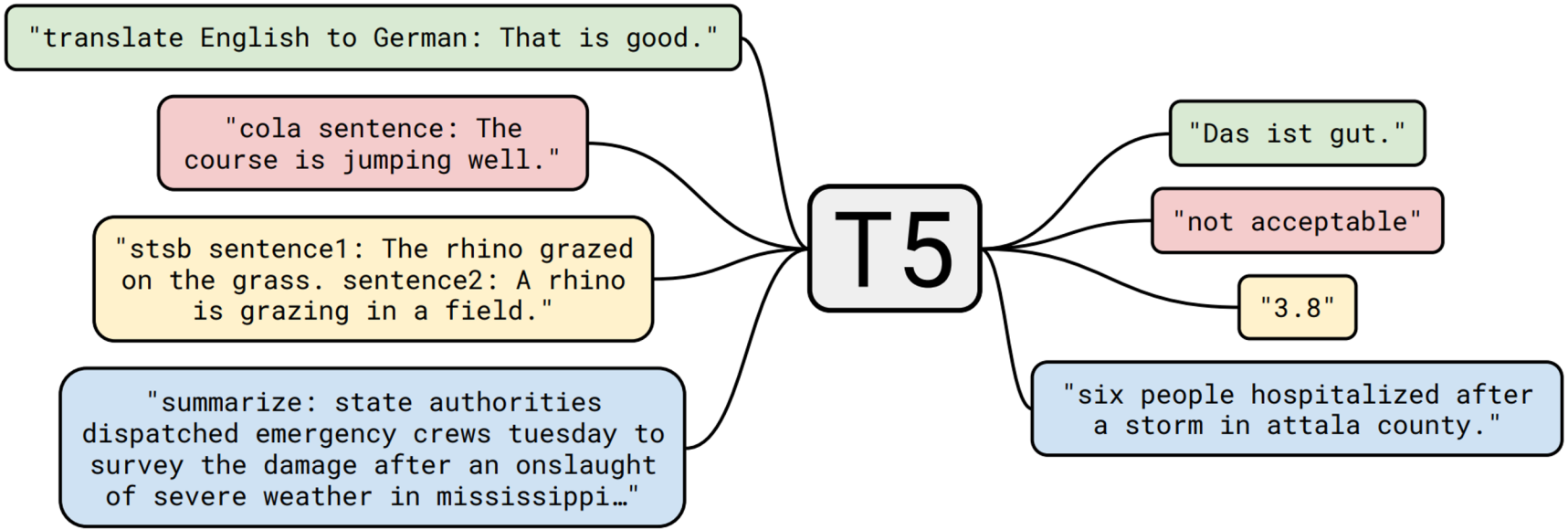
Detection Results (C & C++)

```
void func()
{
    switch(6)
    {
    case 6:
    {
        HCRYPTPROV hCryptProv;
        HCRYPTHASH hHash;
        FILE *pFile = NULL;
        char
password[PASSWORD_INPUT_SIZE];
        UCHAR savedHash[SHA1_SUM_SIZE],
calcHash[SHA1_SUM_SIZE];
        DWORD hashSize;
        char *replace;
        size_t i;
        pFile = fopen("password.txt", "r");
        if (pFile == NULL)
        {
            exit(1);
        }
...
```

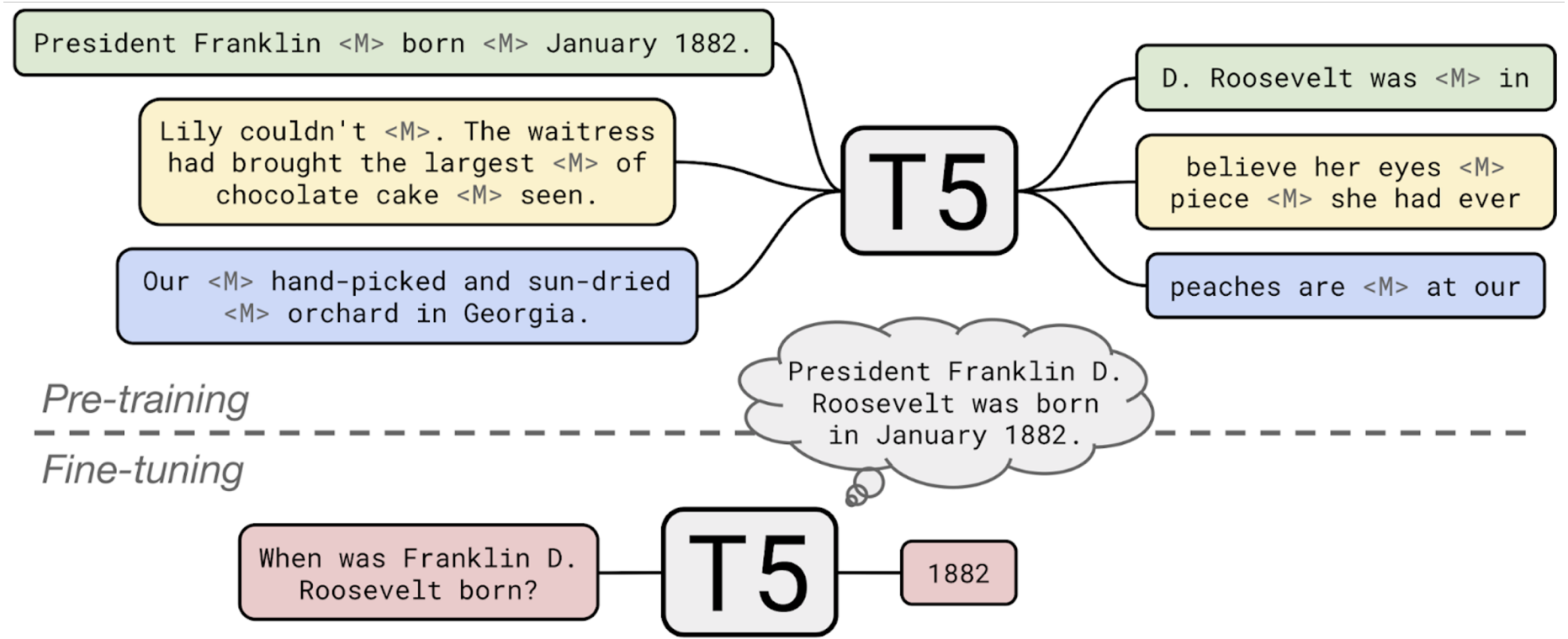


Detection Metrics(Very Good)
SARD: 93%
Draper VDISC: 98% MultiClass(5)
(https://osf.io/d45bw/)

Description Generation with Google's T5 Architecture

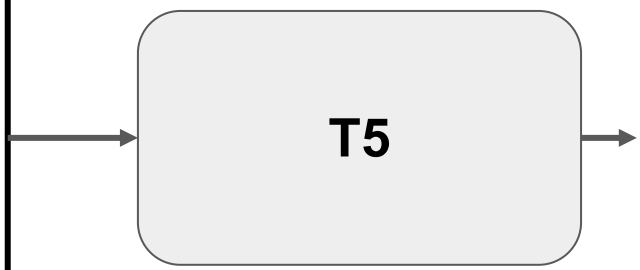


Description Generation with Google's T5 Architecture



Generation Results: Actual Output from Model

```
void func()
{
    switch(6)
    {
    case 6:
    {
        HCRYPTPROV hCryptProv;
        HCRYPTHASH hHash;
        FILE *pFile = NULL;
        char
password[PASSWORD_INPUT_SIZE];
        UCHAR savedHash[SHA1_SUM_SIZE],
calcHash[SHA1_SUM_SIZE];
        DWORD hashSize;
        char *replace;
        size_t i;
        pFile = fopen("password.txt", "r");
        if (pFile == NULL)
        {
            exit(1);
        }
...
```



FLAW: Use a reversible hash (SHA1)
Flaw Located ~line 24

Generation Metrics(Very Good)
Rouge: 0.5942
Bleu: 0.4718

Thanks!

Questions