WEB IR
FINAL PROJECT

HELLO!

TEELEGRAM

B05902013 吳宗翰、B05902021 徐祐謙

B05902086 周　逸、B05902120 曾鈺婷

SOURCE CODE PLAGIARISM

OUTLINE

- Preface (Intro & Difficulty)
- Methodology
- Experiment
- Conclusion

INTRODUCTION

INTRODUCTION

- Retrieval for Word / Article (O)
- Retrieval for Image (O)
- **Retrieval for Code (X)**

→ That's our main idea!

INTRODUCTION

- No one actually do researches related to "query code".
- The most similar topic might be "plagiarism".

METHODOLOGY

FEATURE EXTRACTION

- Similarity computation
  - By plaintext (X)
  - By algorithm / function (O)
- Assembly Code !

ASSEMBLY CODE ?

- Comments Deletion
- Name of Variables
- Solution1 – Optimal Compiler
- Solution2 – Program Behavior

OPTIMAL COMPILER

- -funroll-all-loops
- -fomit-frame-pointer
- -finline-functions
- -fno-stack-protector

OPTIMAL COMPILER

- -fmerge-all-constants
- -ffunction-section
- -fdata-sections
- -mtune=generic

PROGRAM BEHAVIOR

- Type of Instructions (O)
- MemAddress of Instructions (X)
- N-gram (instructions)
- add $0x47000000, %eax => add

OKAPI BM25 MODEL

EXPERIMENT

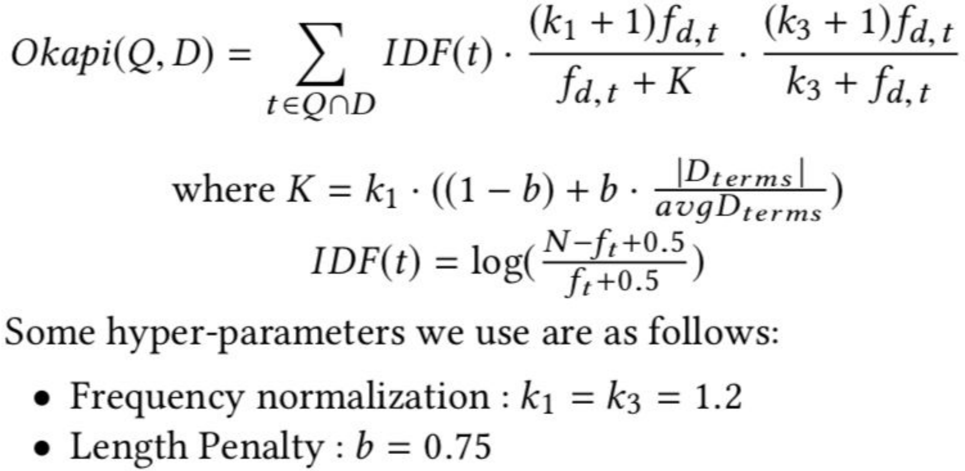
DATA

- Source: Student's Assignment
- Language: C / C++
- Amount: About 8,000 Files
- Ground Truth: MOSS + Human

GOAL

- Query: Those Plagiarism Codes
- To get all plagiarism codes who are based on the query code.

EVALUATION (MAP@10)



$$Okapi(Q,D) = \sum_{t \in Q \cap D} IDF(t) \cdot \frac{(k_1+1) f_{d,t}}{f_{d,t} + K} \cdot \frac{(k_3+1) f_{d,t}}{k_3 + f_{d,t}}$$

$$\text{where } K = k_1 \cdot ((1-b) + b \cdot \frac{|D_{terms}|}{avgD_{terms}})$$

$$IDF(t) = \log(\frac{N - f_t + 0.5}{f_t + 0.5})$$

Some hyper-parameters we use are as follows:

- Frequency normalization : $k_1 = k_3 = 1.2$
- Length Penalty : $b = 0.75$

RESULTS - DIFF COMPILERS

- All Optimal Compilers => Better !
- **“Loop Unrolling”** and **“Function Inline”** => Most Effective !
- Other => Increase a little !

RESULTS - DIFF COMPILERS

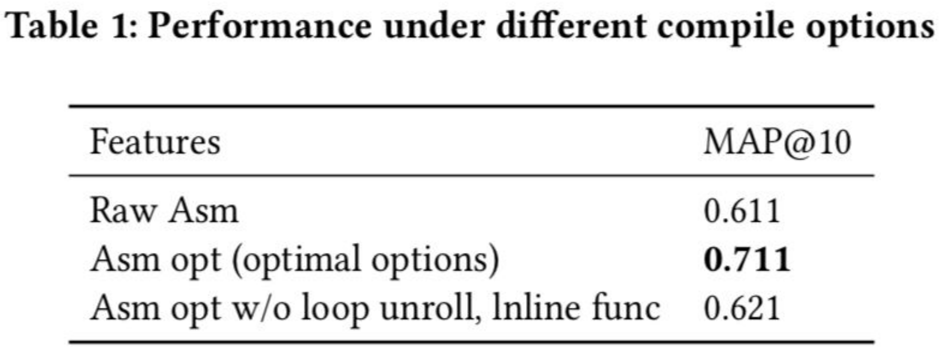


Table 1: Performance under different compile options

Features	MAP@10
Raw Asm	0.611
Asm opt (optimal options)	**0.711**
Asm opt w/o loop unroll, Inline func	0.621

RESULTS - N-GRAM

- Performance similar to the previous methods.
- Unigram => The worst !
- Trigram => The best !

RESULTS - N-GRAM

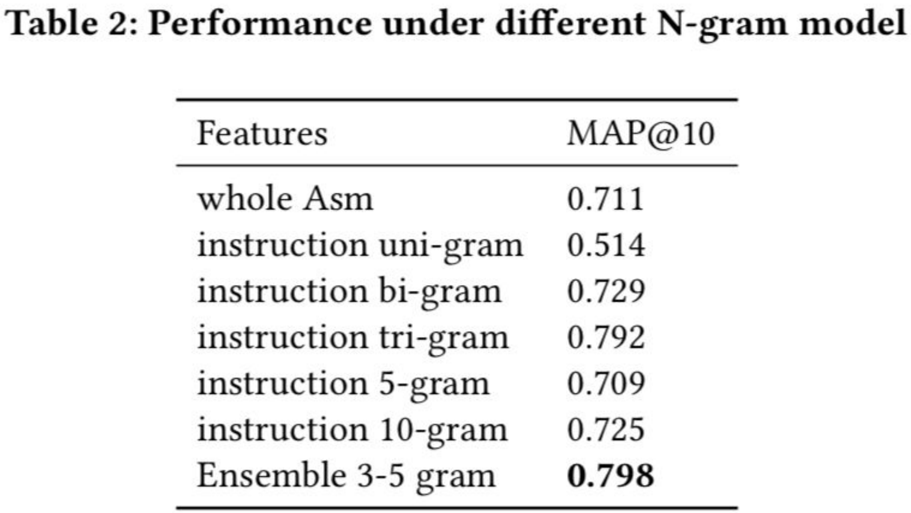


Table 2: Performance under different N-gram model

Features	MAP@10
whole Asm	0.711
instruction uni-gram	0.514
instruction bi-gram	0.729
instruction tri-gram	0.792
instruction 5-gram	0.709
instruction 10-gram	0.725
Ensemble 3-5 gram	**0.798**

RESULTS – PLAGIARISM

- Return top 10 (X)
- Threshold (O)
- Standardization Avg = 0, Var = 1

RESULTS - PLAGIARISM

- Plagiarism (Up to 70%, the score is more than 9)
- Non-Plagiarism (Up to 95%, the score is less than 9)

CONCLUSION

CONCLUSION

- Faster than MOSS (~1 hr)
- Optimal Compiler Options have great influences

CONCLUSION

- N-grams might not be better than single words.
- IR-based Code Plagiarism Detector is feasible.

THANKS!

Any questions?