Plagiarism Detection Tool

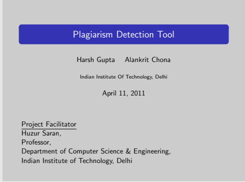
Harsh Gupta Alankrit Chona

Indian Institute Of Technology, Delhi

April 11, 2011

Project Facilitator
Huzur Saran,
Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology, Delhi

Plagiarism Detection Tool


Plagiarism Detection Tool

Harsh Gupta Alankrit Chona

Indian Institute Of Technology, Delhi

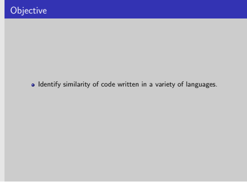
April 11, 2011

Project Facilitator
Huzur Saran,
Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology, Delhi

Harsh

Objective

- Identify similarity of code written in a variety of languages.


Objective
- Identify similarity of code written in a variety of languages.

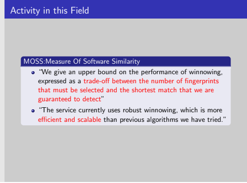
Harsh

Emphasise the importance of such software. In both academic and law enforcement domains

Activity in this Field

MOSS:Measure Of Software Similarity

- “We give an upper bound on the performance of winnowing, expressed as a trade-off between the number of fingerprints that must be selected and the shortest match that we are guaranteed to detect”
- “The service currently uses robust winnowing, which is more efficient and scalable than previous algorithms we have tried.”



Activity in this Field

MOSS:Measure Of Software Similarity

- “We give an upper bound on the performance of winnowing, expressed as a trade-off between the number of fingerprints that must be selected and the shortest match that we are guaranteed to detect”
- “The service currently uses robust winnowing, which is more efficient and scalable than previous algorithms we have tried.”

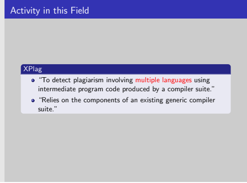
Harsh

- in use for 13 years
- used by most of our professors
- uses hashing algorithm:winnowing which is more efficient and scalable
- They select some hashes to represent a document,calling it the document’s fingerprint and as with any selection criterion, they lose info.’
- emphasis of this algorithm is on efficiency and scalability but we believe that our algo can take into account both the factors without losing any info

Activity in this Field

XPlag

- "To detect plagiarism involving multiple languages using intermediate program code produced by a compiler suite."
- "Relies on the components of an existing generic compiler suite."


Activity in this Field

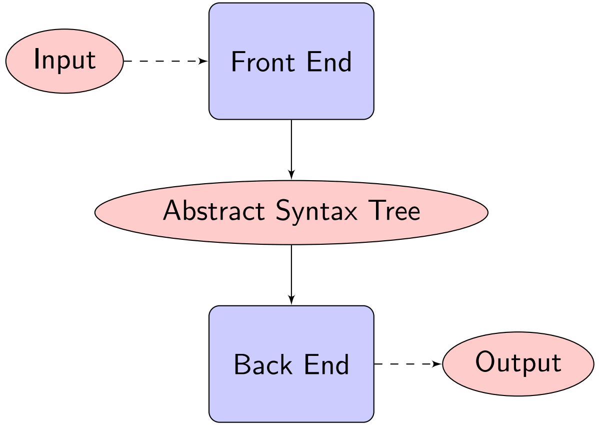
XPlag

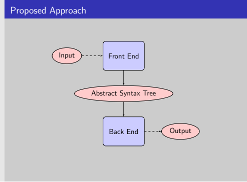
- "To detect plagiarism involving multiple languages using intermediate program code produced by a compiler suite."
- "Relies on the components of an existing generic compiler suite."

Alankrit

- A completely different approach to this problem
- Emphasis on the fact that the same logic can be used in different programming languages

Proposed Approach



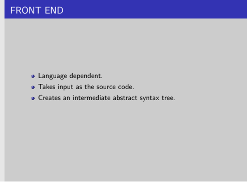


Alankrit

- Input= source code for detection
- FE= scanner and CUSTOM parser for the input language
- AST= a generic AST which will be similar for all languages
- BE= the computation intensive matching algorithm
- Results: matching pairs

FRONT END

- Language dependent.
- Takes input as the source code.
- Creates an intermediate abstract syntax tree.



FRONT END

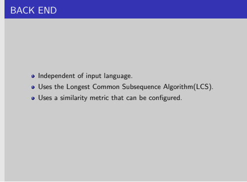
- Language dependent.
- Takes input as the source code.
- Creates an intermediate abstract syntax tree.

Harsh

- implementation of a lexer and a CUSTOM parser for the language
- emphasize that no significant syntactical information is lost .
- point out that this information will be used throughout the entire computation.

BACK END

- Independent of input language.
- Uses the Longest Common Subsequence Algorithm(LCS).
- Uses a similarity metric that can be configured.

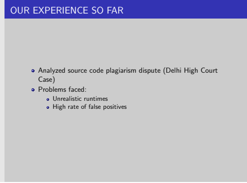

BACK END
- Independent of input language.
- Uses the Longest Common Subsequence Algorithm(LCS).
- Uses a similarity metric that can be configured.

Harsh

- while independent of the input language,syntactical information will be retained and used throughout the code
- use of LCS algo to check for similarity
- use of LCS algo in the diff program commonly used in versioning systems like git
- the similarity metric can be configured to provide for a loose or strict metric.

OUR EXPERIENCE SO FAR

- Analyzed source code plagiarism dispute (Delhi High Court Case)
- Problems faced:
  - Unrealistic runtimes
  - High rate of false positives



OUR EXPERIENCE SO FAR

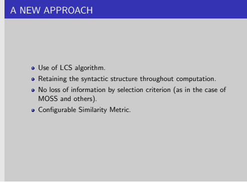
- Analyzed source code plagiarism dispute (Delhi High Court Case)
- Problems faced:
  - Unrealistic runtimes
  - High rate of false positives

Alankrit

- stress the fact that we have learnt the importance of intelligently applying LCS rather than a greedy algo
- backtracking,
  - assuming a root and its children to be a match and then verifying correctness of assumption
  - increasing tightness of similarity metric as we move down the tree

A NEW APPROACH

- Use of LCS algorithm.
- Retaining the syntactic structure throughout computation.
- No loss of information by selection criterion (as in the case of MOSS and others).
- Configurable Similarity Metric.


A NEW APPROACH

- Use of LCS algorithm.
- Retaining the syntactic structure throughout computation.
- No loss of information by selection criterion (as in the case of MOSS and others).
- Configurable Similarity Metric.

Harsh

- we like the lcs algo and wonder why people have not used it before
- striking difference from others: no loss of syntactical information to the last stage of computation
- by appling LCS intelligently we can ensure that no loss of info is incurred and runtimes are reasonable
- configurable system will allow it to be used in variety of enviroments–academic, legal etc.

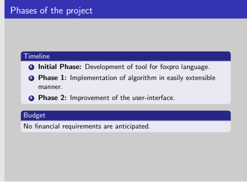
Phases of the project

Timeline

1. **Initial Phase:** Development of tool for foxpro language.
2. **Phase 1:** Implementation of algorithm in easily extensible manner.
3. **Phase 2:** Improvement of the user-interface.

Budget

No financial requirements are anticipated.



Phases of the project

Timeline

1. Initial Phase: Development of tool for foxpro language.
2. Phase 1: Implementation of algorithm in easily extensible manner.
3. Phase 2: Improvement of the user-interface.

Budget

No financial requirements are anticipated.

Thank You