



# Plagiarism Detection In Programming Language Source Codes Using NLP Tree Kernel Methods

**Group - 27**

**Mentors:**

Amrith Krishna  
Binny Mathew

**Members:**

Ashish Manmode	12EC35006
Bishal Santra	12EC35001
Deep Kiran Shroti	12EC35013
Eashaan Baberwal	12EC35016
Vishnu Dutt Sharma	12EC35013
Soumya Sanyal	12EC10056

# INTRODUCTION

- Academic dishonesty is a universal problem (our institute included)
- Methods for finding duplicate code among peers generally works with heuristics based work or other language model based methods, which has proven to be ineffective in current scenario e.g. MOSS, JPlag, Sherlock
- Advantage of a computer program, over normal text: Well defined formal grammar
- Source codes are easy to process with the help of Abstract Syntax Tree and other flow diagrams using compiler toolchains.

# LIBRARIES USED

- CLANG

- **Clang** is a compiler front end for the C, C++, Objective-C and Objective-C++ programming languages. It uses LLVM as its back end and has been part of the LLVM release cycle since LLVM 2.6.

- NLTK ( <http://www.nltk.org/> )

- NLTK is a leading platform for building Python programs to work with human language data.

- TREE-SVM ( <https://github.com/sitfoxfly/tree-svm> )

- Tree-SVM is used for Sub Tree, Subset Tree & Partial Tree Kernel extraction.

- Scikit-learn ( <http://scikit-learn.org/stable/> )

- Scikit-learn is used to apply SVM.





## DATASET USED

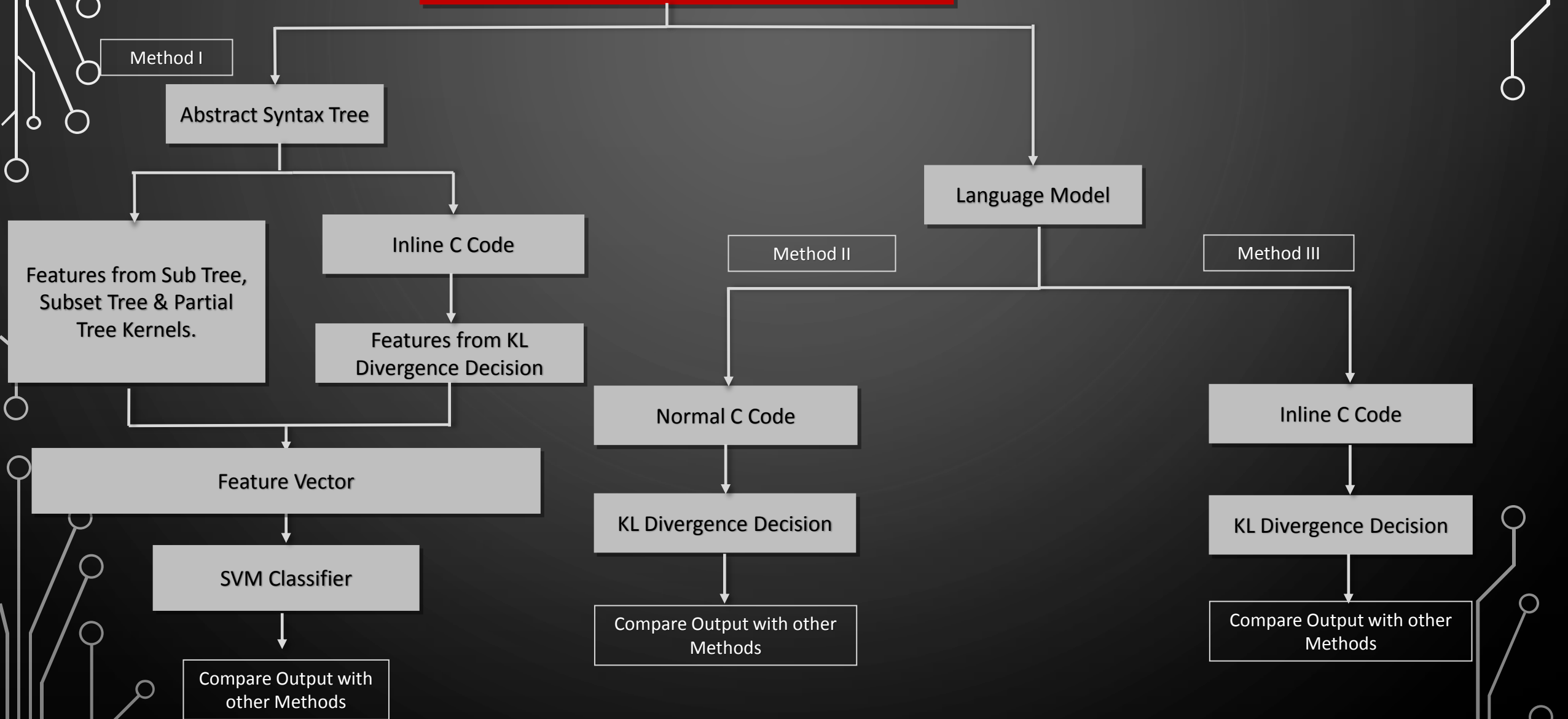
- A set of C source code files, each containing multiple files for each user provided by mentors.
- Used MOSS (Measure Of Software Similarity) to assign tags to the set of plagiarized codes.

# CONCEPTS USED

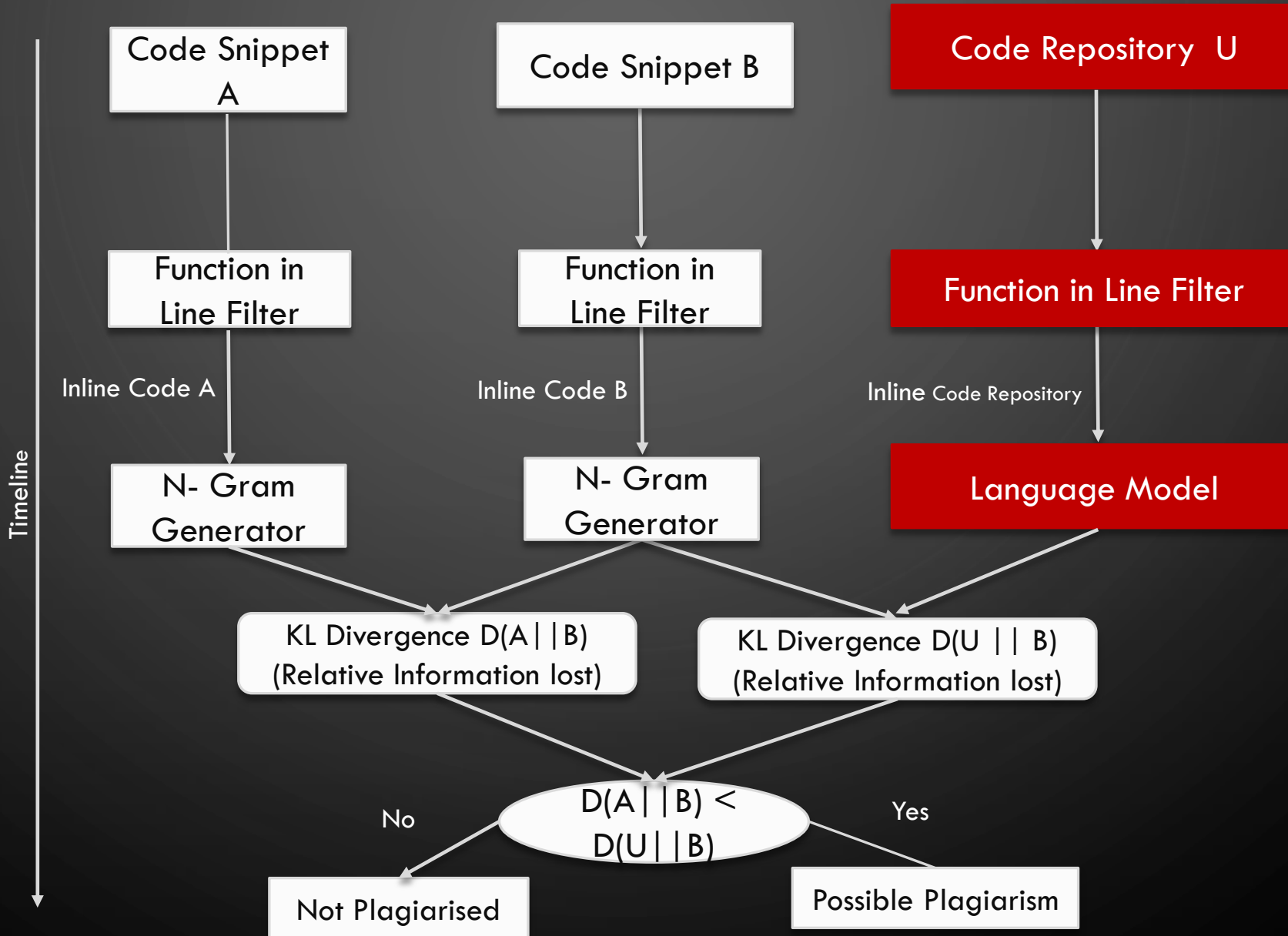
- Language Model
  - Statistical Language Modelling is to build a statistical language model that can estimate the distribution of natural language as accurate as possible.
  - In-lining has been done to catch plagiarism, even if a person makes separate function for various parts of the code.
- AST transversal
  - An abstract syntax tree is a tree representation of the abstract syntactic structure of the source code. Each node of the tree denotes a construct occurring in the source code.
- KL Divergence
  - The Kullback–Leibler divergence, also known as information divergence, relative entropy is a non-symmetric measure of the difference between two probability distributions  $P$  and  $Q$ .

# PLAGIARISM DETECTION MODEL

Plagiarized Source Code Data Set



# THE PLAGIARISM DETECTION ARCHITECTURE USING LANGUAGE MODEL



# APPROACH

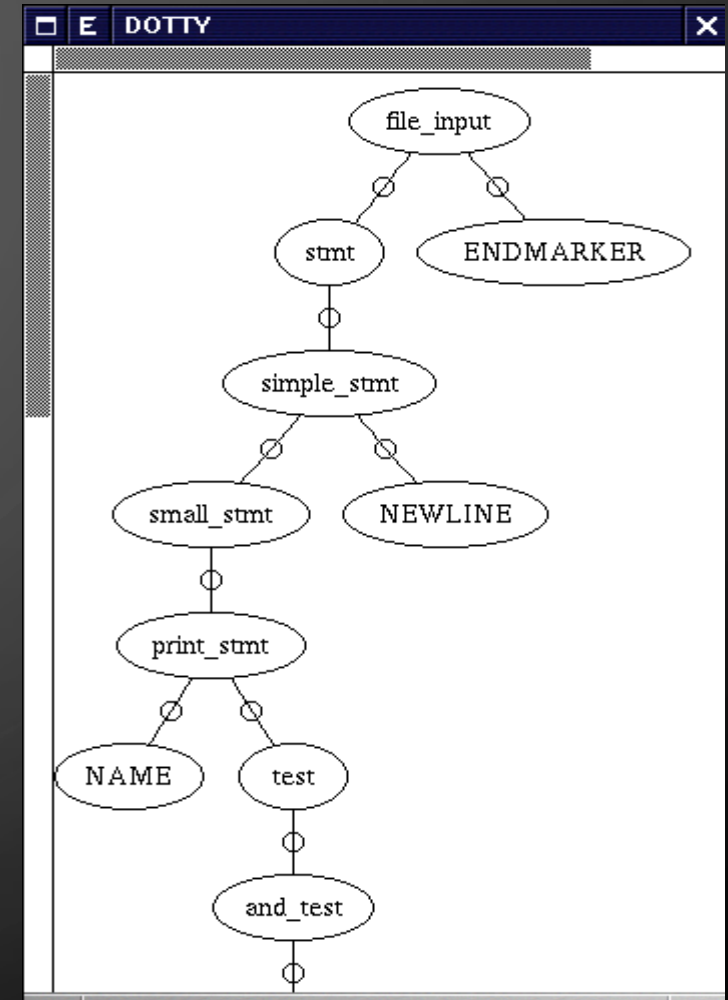
- Language Model

- N-grams (up to trigrams) keyword comparison for C source codes using KL divergence. For bigrams matching technique, we need to consider the possibility that in a plagiarised code, a function of original code may have been written after in-lining. So before extracting keywords, we need to replace all function references in the codes with respective function declarations.



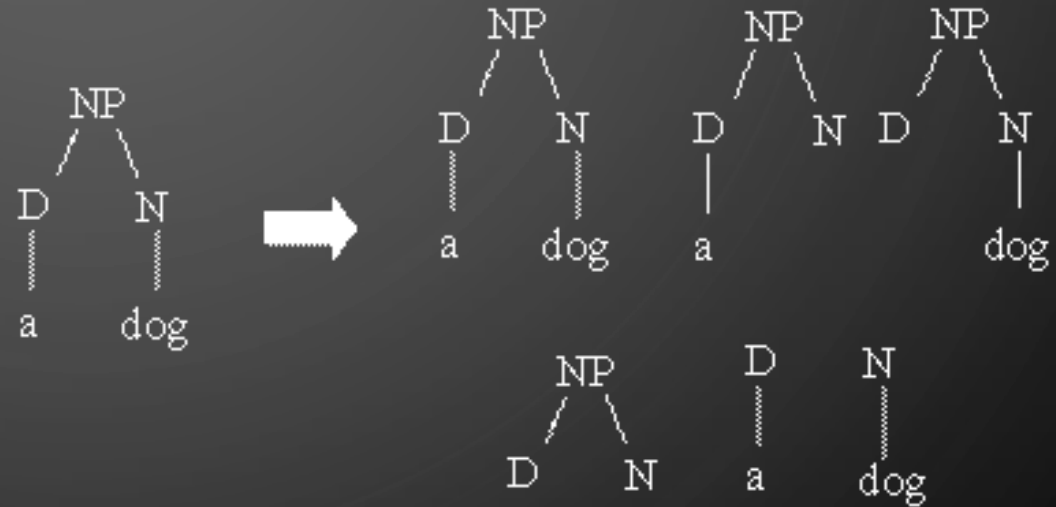
## Abstract Syntax Tree Matching

- Abstract Syntax Tree - It provides the details about the basic structure of a code, such as Function declaration and Identifiers
- Generated using '*clang*'
- Also helps in avoiding the attempt of deceiving system by changing variable or function name



# WHAT IS NLP TREE KERNEL?

- Think of Kernel as a transformation
- Convert to the basic components
- More suitable for source codes as generation of trees is easier using compiler toolchains



# TYPES OF KERNEL USED

## 1. Subtree Kernel (ST Kernel)

- The kernel returns a weighted sum of the number of common substrings (proper subtrees)

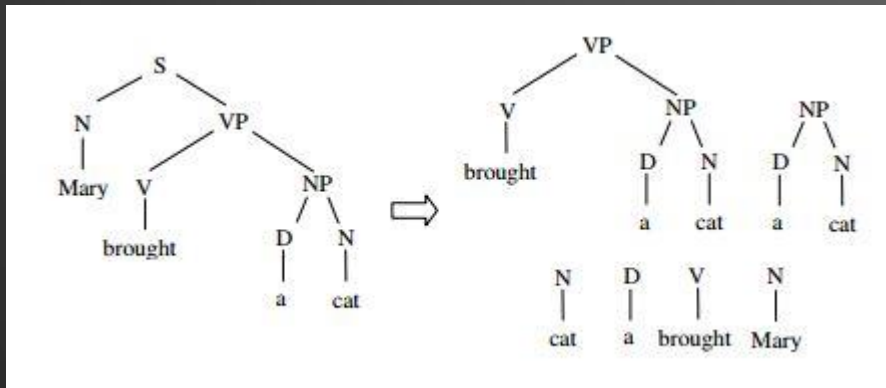
## 2. Subset Tree Kernel (SST Kernel)

- The subset tree kernel (SST) defines a similarity measure between trees which is proportional to the number of shared subset trees

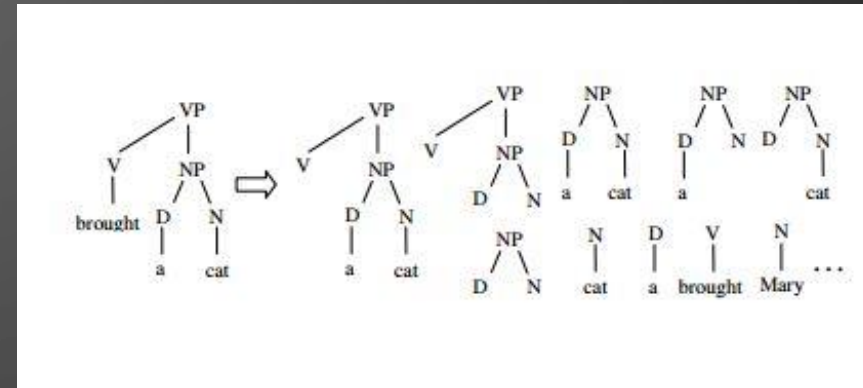
## 3. Partial Tree Kernel (PT Kernel)

- A convolutional tree kernel, obtained by relaxing the constraint over SSTs that grammatical rules can't be broken

# TREE KERNELS EXAMPLES

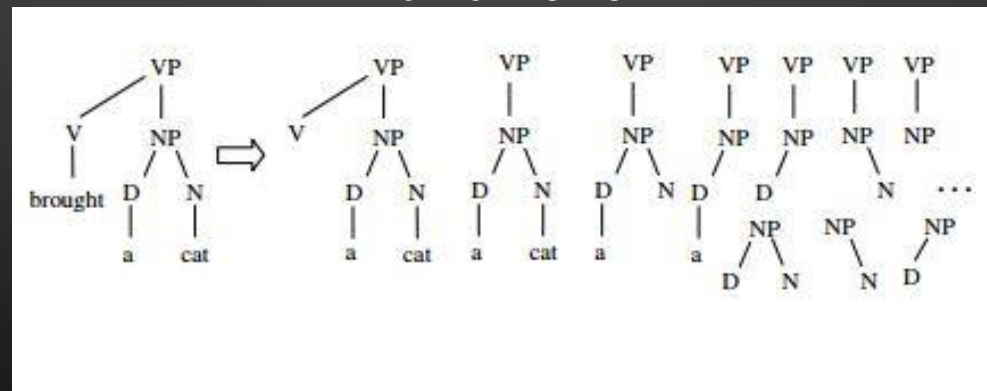


Subtree Kernel

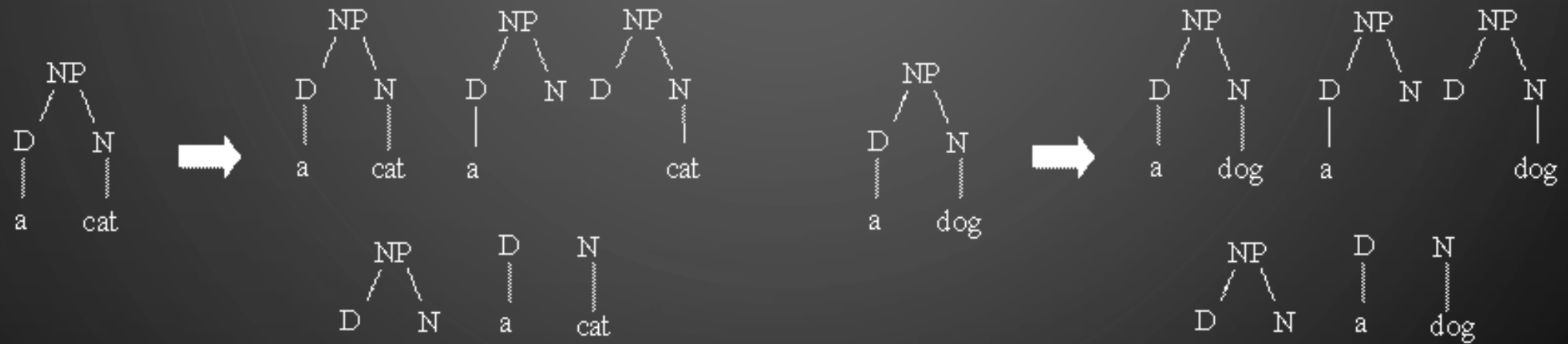


Subset Tree Kernel

Partial Kernel



# KERNEL MATCHING: EXAMPLE



As 3 structures (out of 5) are completely identical the similarity is equal to 3

# CLASSIFICATION USING SVM

1. Feature Vector: The 6 dimensional feature vector used for SVM contains 3 Kernel Similarity measures (Subset Tree, Sub Tree, Partial Tree) and KL Divergence between (File1 and File2), (File2 and GeneralModel), (File2 and GeneralModel) between each pairs of files.

# CONFUSION MATRIX

	True Prediction	False Prediction	Total
Actual True Data (Plagiarism)	True Positive = 20	False Negative = 3	23
Actual False Data (No Plagiarism)	False Positive = 13	True Negative = 117	130
Total	33	120	

# RESULTS

## Precision and Recall

1. Precision = true positives / (true positives + false positives) =  $20 / (20 + 13) = 60\%$
2. Recall = true positives / (true positives + false negatives) =  $20 / (20 + 3) = 87\%$

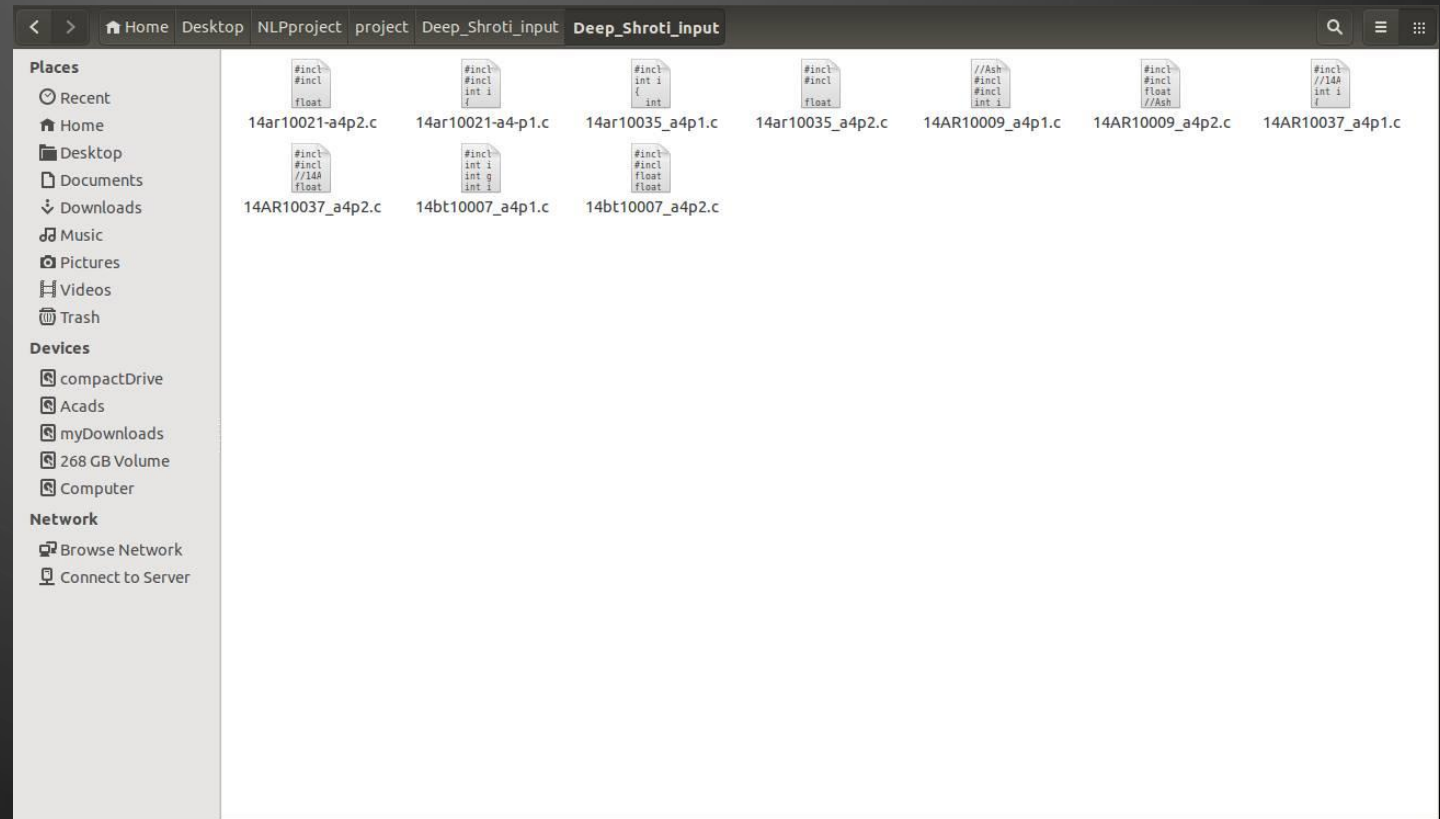
## Comparison against baseline

- Applying SVM on kl divergence gives 50% accuracy on 10 fold cross validation.
- Whereas SVM on kl divergence with Kernel Tree gives 78% accuracy on 10 fold cross validation.



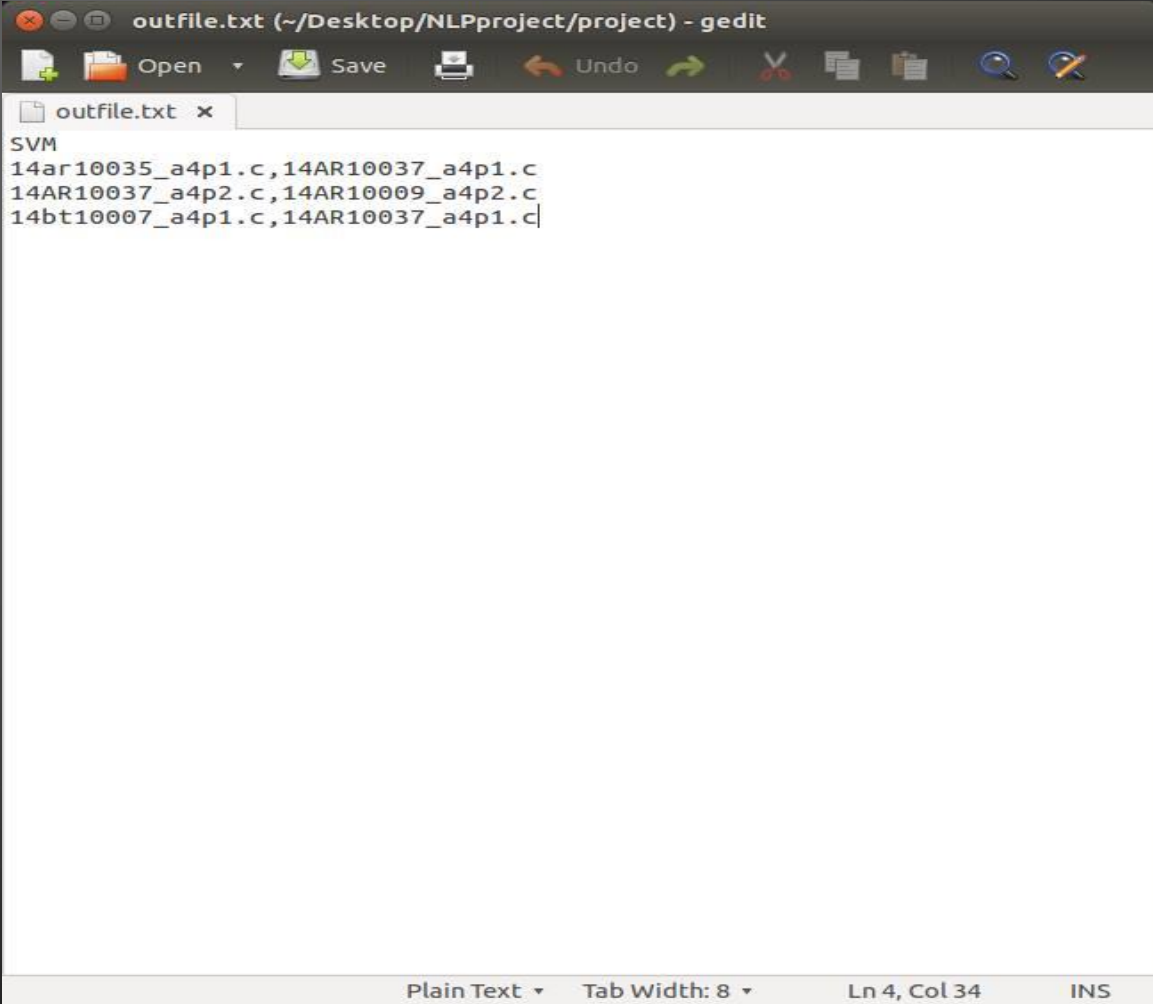
# INPUT FORMAT

- The user needs to paste the source code files to check for plagiarism detection in a specified folder as shown in the figure.



# OUTPUT FORMAT

The pair of plagiarised codes will be shown in an output.txt file created in the project folder.



```
outfile.txt (~/Desktop/NLPproject/project) - gedit
SVM
14ar10035_a4p1.c,14AR10037_a4p1.c
14AR10037_a4p2.c,14AR10009_a4p2.c
14bt10007_a4p1.c,14AR10037_a4p1.c
```

The screenshot shows a gedit window titled "outfile.txt (~/Desktop/NLPproject/project) - gedit". The window contains the following text:

```
SVM
14ar10035_a4p1.c,14AR10037_a4p1.c
14AR10037_a4p2.c,14AR10009_a4p2.c
14bt10007_a4p1.c,14AR10037_a4p1.c
```

The status bar at the bottom of the window indicates "Plain Text", "Tab Width: 8", "Ln 4, Col 34", and "INS".

# ACKNOWLEDGEMENT

- We would like to take this opportunity to extend our thanks and gratitude to Professor Pawan Goyal for his guidance and encouragement throughout the course of our term project. His encouragement for finding new ideas to work on and choosing our own topic helped us gain interest and confidence in this topic. We are indebted to him for giving us this exposure in the field of Natural Language Processing.
- We would also like to thank our mentors, Amrith Krishna and Binny Mathew for initiating our interest in NLP and their constant guidance, help and support over the last few months in finding and learning new things in this field.



# CITATIONS

- M. Tkachenko and H. W. Lauw. A convolution kernel approach to identifying comparisons in text. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2015.

# BIBLIOGRAPHY

- R.-E. Fan, P.-H. Chen, and C.-J. Lin, “Working set selection using second order information for training support vector machines,” *J. Mach. Learn. Res.*, vol. 6, pp. 1889–1918, Dec. 2005.
- M. Collins and N. Duffy, “New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron,” in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, (Philadelphia, Pennsylvania, USA), pp. 263–270, Association for Computational Linguistics, July 2002.
- A. J. Smola and S. Vishwanathan, “Fast kernels for string and tree matching,” in *Advances in Neural Information Processing Systems 15* (S. Becker, S. Thrun, and K. Obermayer, eds.), pp. 585–592, MIT Press, 2003.
- A. Moschitti, “Efficient convolution kernels for dependency and constituent syntactic trees,” in *Machine Learning: ECML 2006* (J. Frnkranz, T. Scheffer, and M. Spiliopoulou, eds.), vol. 4212 of *Lecture Notes in Computer Science*, pp. 318–329, Springer Berlin Heidelberg, 2006.

# REFERENCE

- Alberto BarronCedeno, Paolo Rosso, and JoseMiguel Benede, Reducing the Plagiarism Detection Search Space on the Basis of the KullbackLeibler Distance
- Thomas Bakker, W. A. Kusters, E. A. Verbitskiy, Plagiarism Detection in Source Code
- Asim M. El Tahir Ali, Hussam M. Dahwa Abdulla and Vaclav Snasel, Overview and Comparison of Plagiarism Detection Tools
- Matt G. Ellis, Claude W. Anderson, Plagiarism Detection in Computer Code
- Alberto Barron-Cedeno and Paolo Rosso, On Automatic Plagiarism Detection Based on nGrams Comparison
- KullbackLeibler Divergence, available at <https://en.wikipedia.org/wiki/Kullback>
- Language Model, available at [https://en.wikipedia.org/wiki/Language Model](https://en.wikipedia.org/wiki/Language_Model)



THANK YOU