Εξαγωγή Προτύπων Αλλαγών Κώδικα από Αποθετήρια Ανοικτού Λογισμικού

Εκπόνηση:

Οδυσσέας Κυπαρίσσης

Α.Ε.Μ: 8955

Επίβλεψη:

Αν. Καθηγητής, Ανδρέας Συμεωνίδης

Υπ. Διδάκτωρ, Θωμάς Καρανικιώτης

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Τομέας Ηλεκτρονικής και Υπολογιστών

Ομάδα Ευφυών Συστημάτων και Τεχνολογίας Λογισμικού (ISSEL)

Διάρθρωση Παρουσίασης

- **Εισαγωγή**
- **Μεθοδολογία**
- **Αξιολόγηση & Αποτελέσματα**
- **Συμπεράσματα**
- **Μελλοντική Εργασία**
- **Ευχαριστίες**

Εισαγωγή

Περιγραφή Προβλήματος:

- Επανειλημμένη αντιμετώπιση παρόμοιων σφαλμάτων πηγαίου κώδικα από διαφορετικές ομάδες προγραμματιστών.
- Ελάχιστη προσπάθεια για ανάπτυξη επαναχρησιμοποιήσιμου λογισμικού.
- Αυξανόμενη ανάγκη για γρήγορη και αποτελεσματική ανάπτυξη πηγαίου κώδικα.

Εισαγωγή

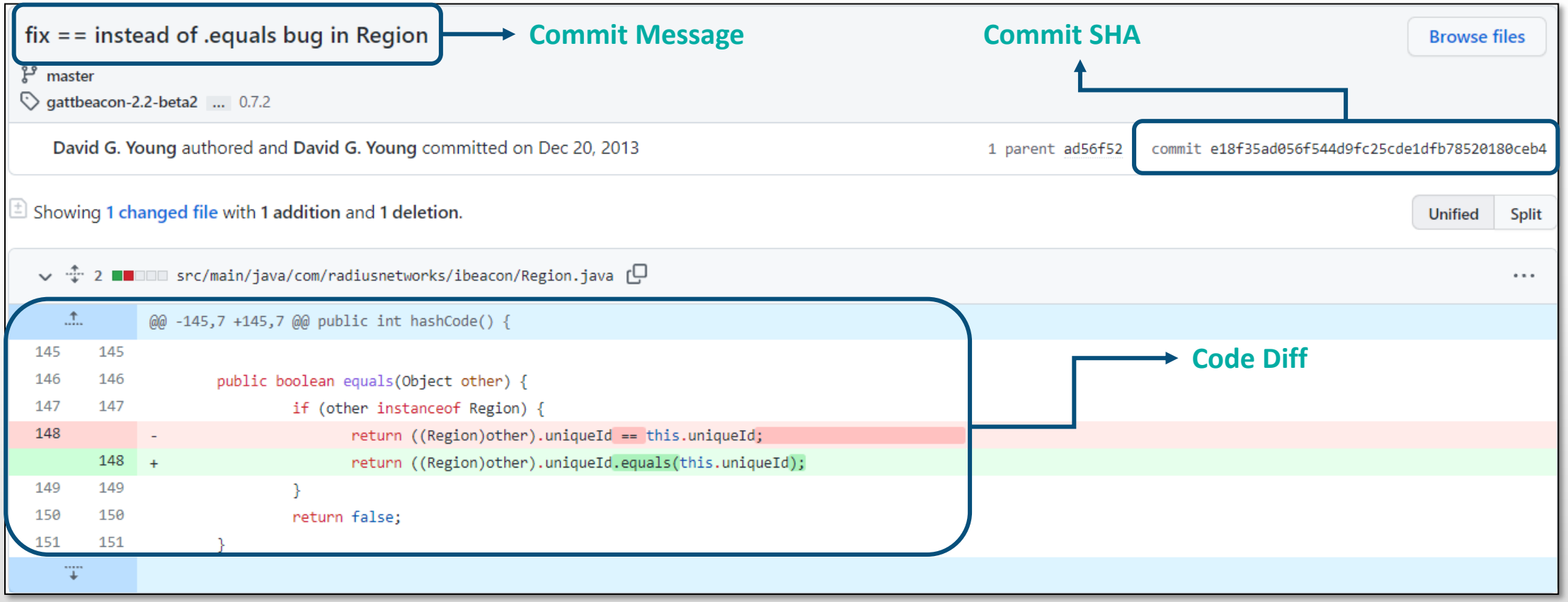
Σκοπός Διπλωματικής:

Ανάπτυξη αυτοματοποιημένου συστήματος εξόρυξης προτύπων αλλαγών από αποθετήρια ανοικτού λογισμικού.

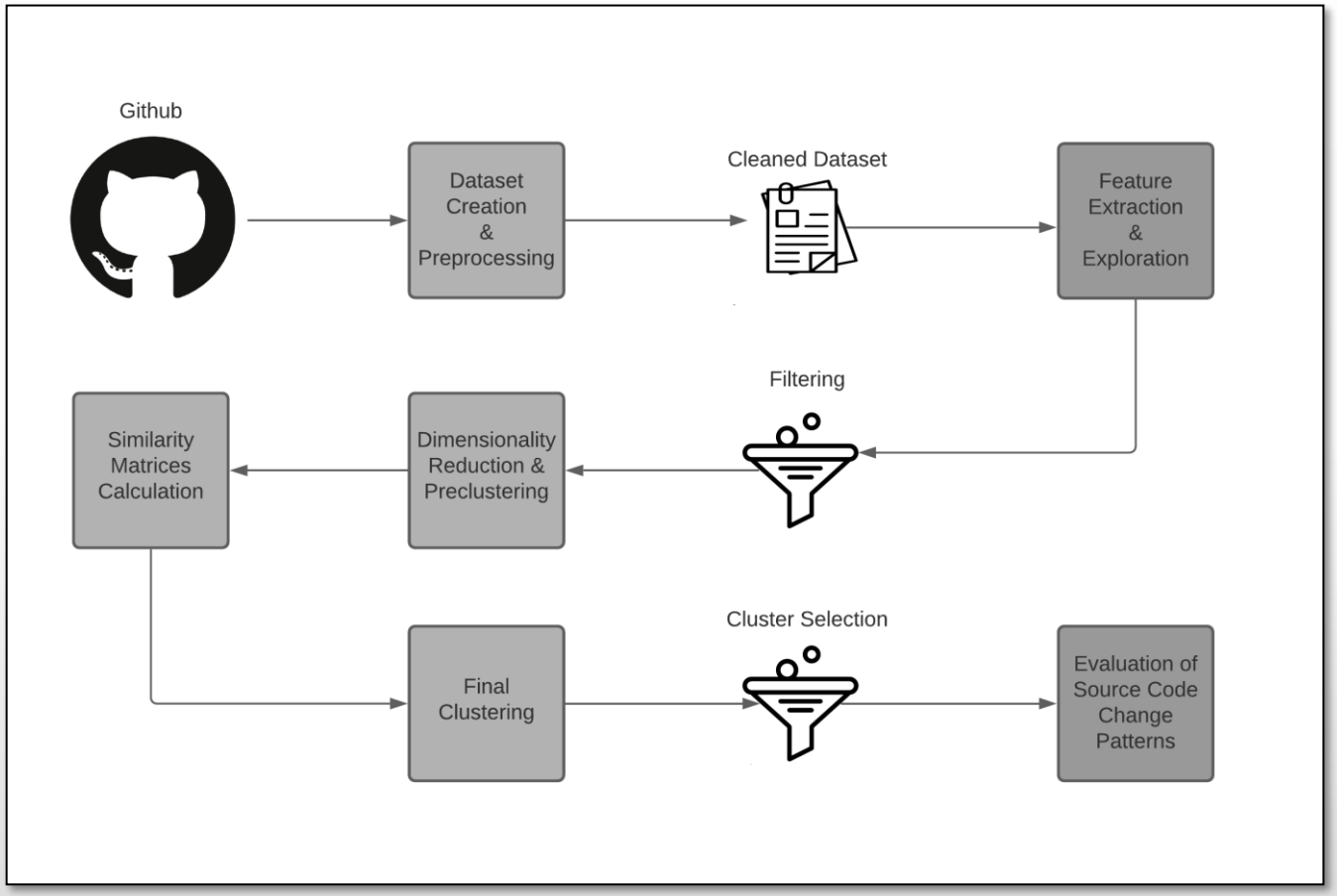
- Μελέτη εξέλιξης τμημάτων πηγαίου κώδικα μεγάλων έργων λογισμικού.
- Εξαγωγή και πρόταση γενικευμένων προτύπων αλλαγών πηγαίου κώδικα με στόχο:
  - Την αποσφαλμάτωση λογισμικού.
  - Την μείωση του χρόνου αποσφαλμάτωσης κατά την ανάπτυξη λογισμικού.
  - Την επαναχρησιμοποίηση πηγαίου κώδικα.

Εισαγωγή

Βασική Μονάδα Ανάλυσης - GitHub Commit:



Μεθοδολογία



Μεθοδολογία

Δημιουργία & Προεπεξεργασία Συνόλου Δεδομένων

- 900 δημοφιλέστερα Java αποθετήρια του GitHub:
  - 600 → Training Set
  - 300 → Test Set
- Επιλογή αποθετηρίων με < 2500 commits.
- Επιλογή commits μόνο του main κλάδου.
- Απόρριψη commits ⇒ message ≠:
  - fix, improve, change, bug, add, remove, support.
- Διαχωρισμός των code changes σε 3 κατηγορίες.
  - Both, Only Additions, Only Deletions.

137.112 – 73.15%

35.490 – 18.93%

14.839 – 7.92%

730.320 Code Changes

Διάσπαση Commits Πολλαπλών Αρχείων

- **453.257** Code Changes
- **62%**

Διαγραφή Αλλαγών Κύριου Τμήματος (main)

- **89.622** Duplicates
- **12.3%**

Διαγραφή Διπλοεγγραφών Ίδιου Commit

187.441 Code Changes

Μεθοδολογία

Τελικά Χαρακτηριστικά

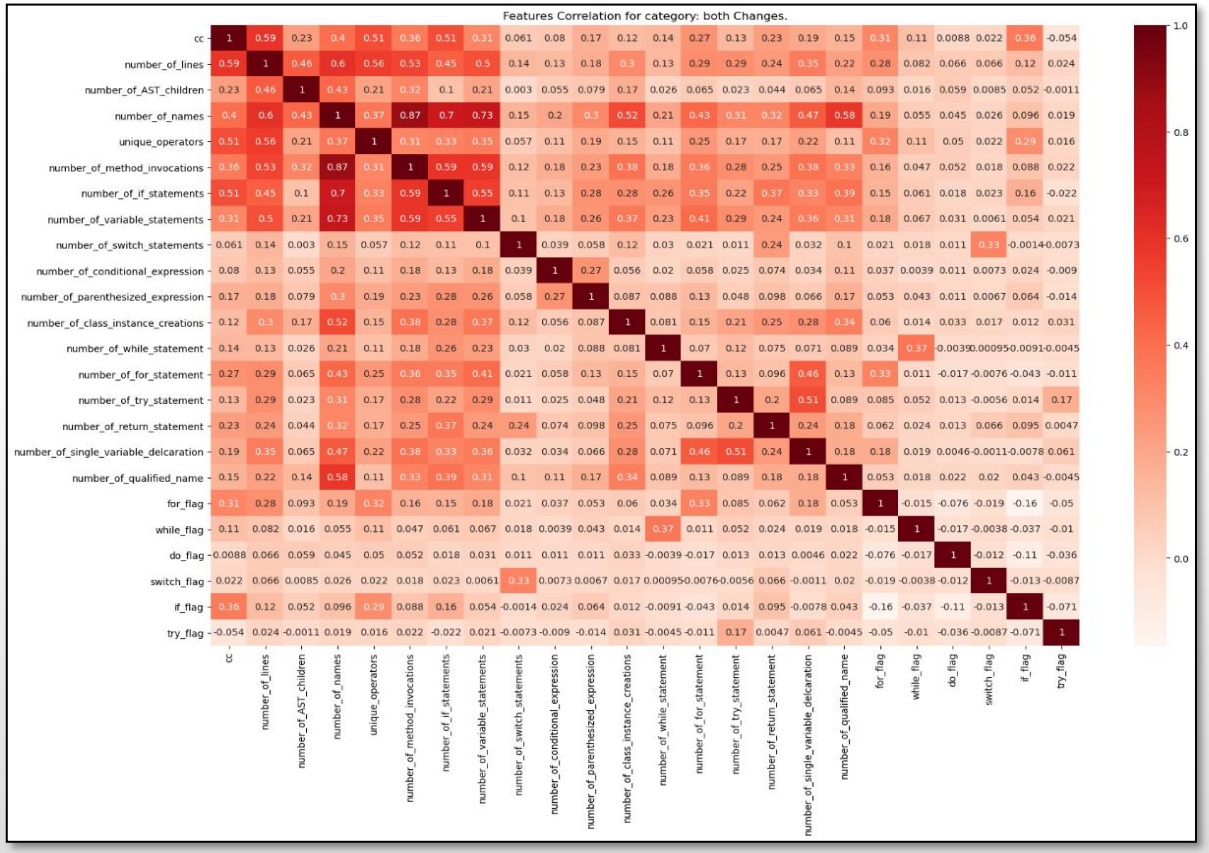
- Μετατροπή πηγαίου κώδικα σε Αφηρημένο Συντακτικό Δέντρο (AST).
- Τελικά Χαρακτηριστικά του Συνόλου Δεδομένων:

Cleaned Dataset										
SHA	Message	Code Diff	Method Code Before	Method Code After	Method Code Before AST	Method Code After AST	Code Additions	Code Deletions	Code Additions AST	Code Deletions AST

Μεθοδολογία

Εξαγωγή & Ανάλυση Χαρακτηριστικών

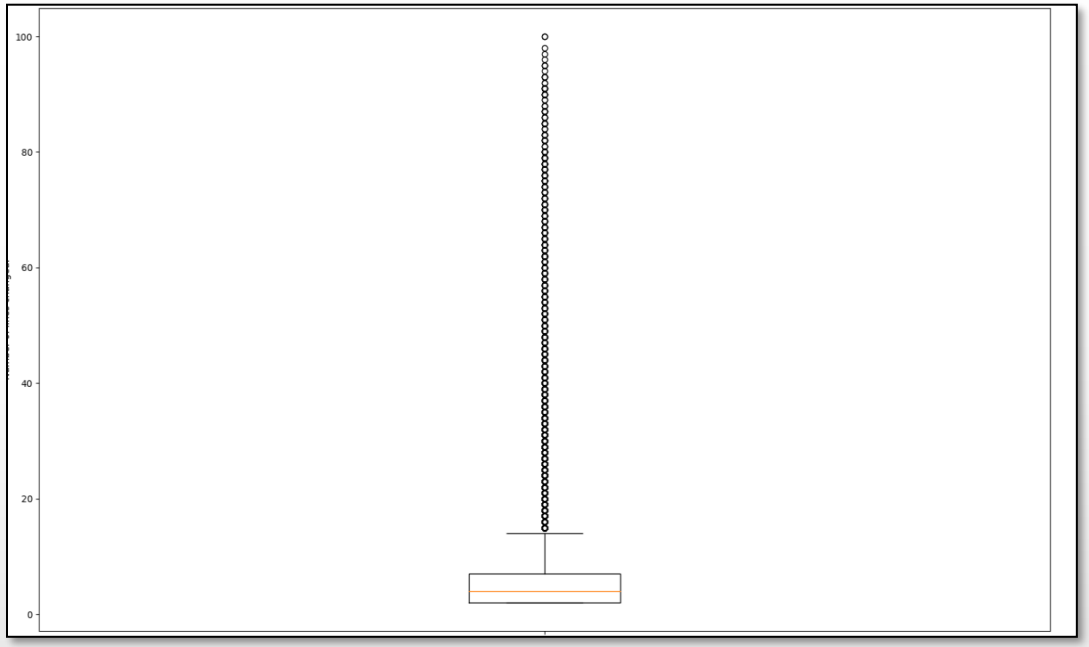
Χαρακτηριστικά (#) :	
Lines Changed	Variable Statements
Cyclomatic Complexity	Switch Statements
Unique Operators	Simple Names
AST Children	Class Instance Creations
Conditional Expressions	While Statements
Method Invocations	For Statements
If Statements	Try Statements
Return Statements	Qualified Names
Single Variable Declaration	



Μεθοδολογία

Φιλτράρισμα & Μείωση Διαστασιμότητας

Box Plot – Number of Lines – Both Category



Category	Size Before	Filtered	Percentage
Both	137.112	18.647	13,6%
Only Additions	35.490	4,950	13,95%
Only Deletions	14.839	1.700	11,46%

Feature	Code Changes Category	
	Both	**Only Additions & Deletions**
Number of lines >	16	8
Number of AST children >	14	6
Number of names >	41	20

- Εφαρμογή Αλγορίθμου Μείωσης Διαστασιμότητας
  - Principal Component Analysis – **PCA**.
  - Μετασχηματισμός των δεδομένων από **22** διαστάσεις σε **10**.
  - Περιλαμβάνουν το **95%** της συνολικής πληροφορίας και για τις τρείς κατηγορίες.

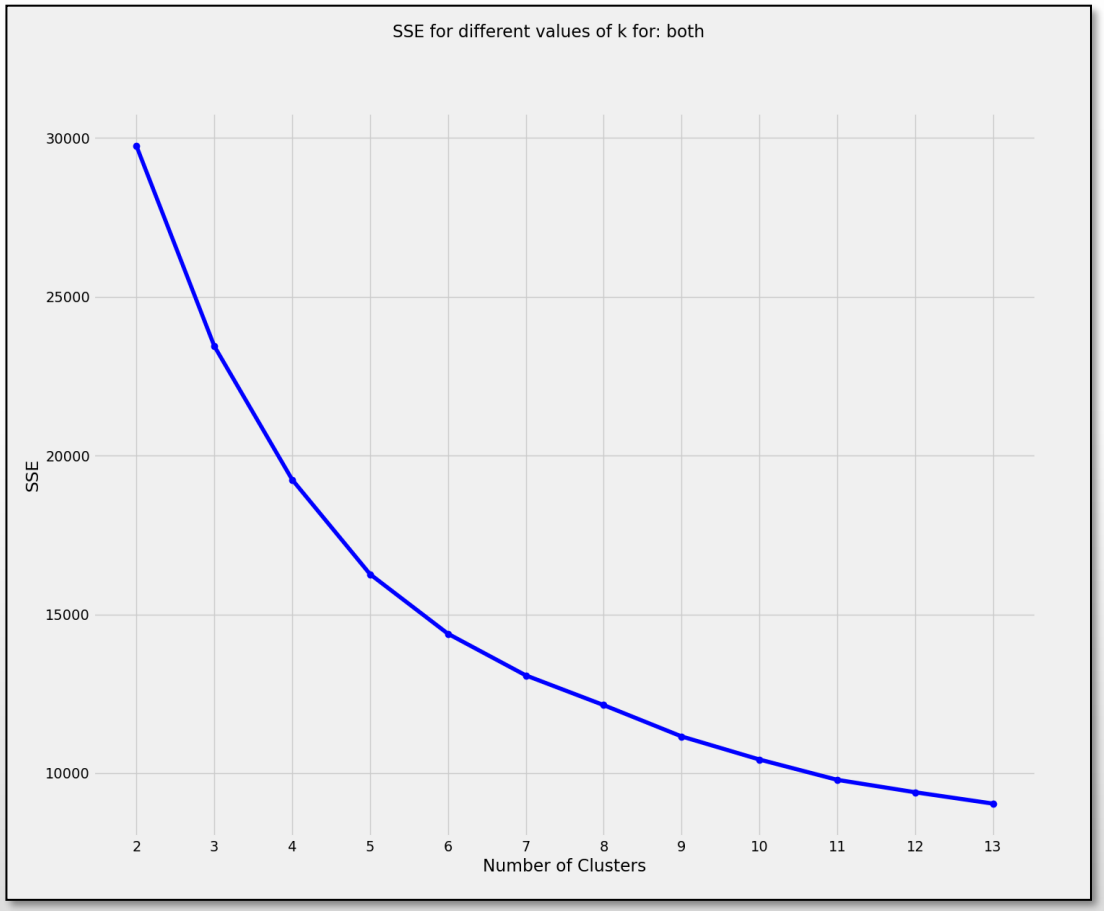
Μεθοδολογία

Αρχική Ομαδοποίηση

- Εφαρμογή Αλγορίθμου Ομαδοποίησης **k-means ++.**
- Υπολογισμός του αθροίσματος τετραγωνικού σφάλματος για την επιλογή του **k**.

$$\boldsymbol{SSE} = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x)$$

Data Category	Number of Clusters
Both	9
Only Additions	5
Only Deletions	5

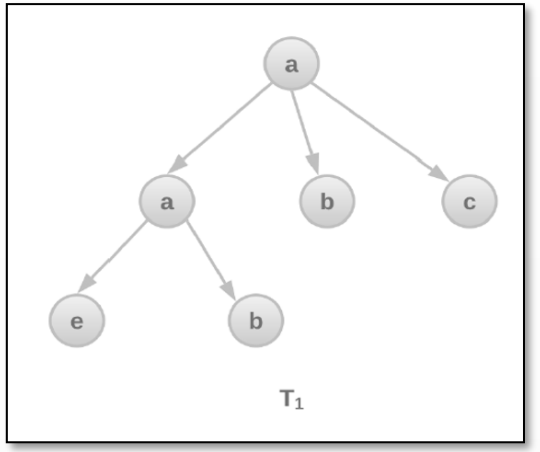


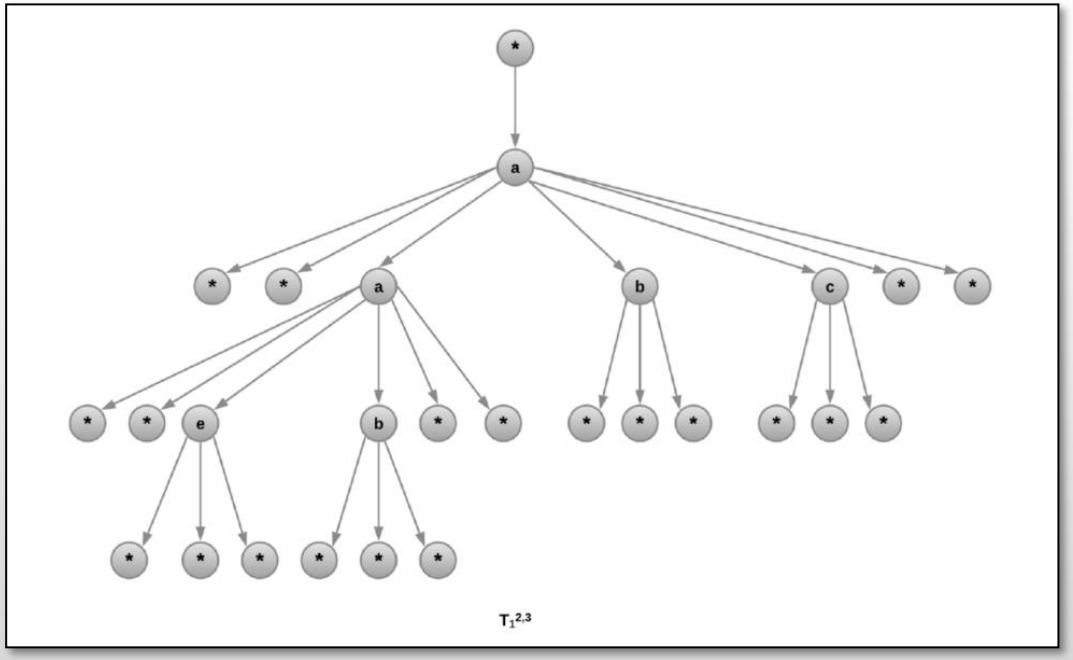
Μεθοδολογία

Υπολογισμός Πινάκων Ομοιότητας Αρχικών Ομάδων

Ομοιότητα αλγοριθμικής δομής των αλλαγών.

- Χρήση αλγορίθμου TED (Tree Edit Distance) – **pq-grams**.
- Βασίζεται σε δύο παραμέτρους $\boldsymbol{p}, \boldsymbol{q}$.
- Για κάθε ένα AST σχηματίζεται ένα Ordered Labeled Tree.
  - $(p-1)$ null κόμβοι εισάγονται στη ρίζα.
  - $(q-1)$ null κόμβοι εισάγονται, πριν το πρώτο και μετά το τελευταίο παιδί, κάθε ενδιάμεσου κόμβου (non-leaf node).
  - q παιδιά εισάγονται σε κάθε φύλλο του δέντρου.



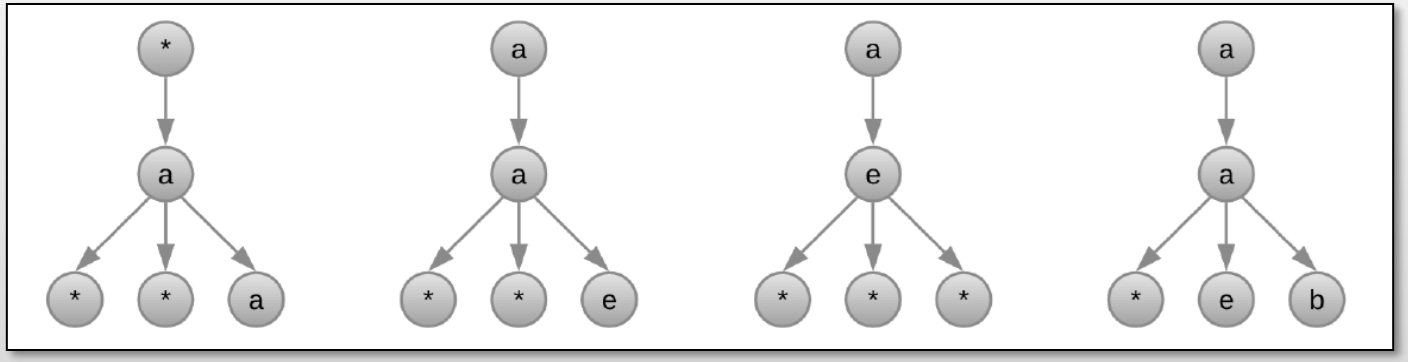


Μεθοδολογία

Υπολογισμός Πινάκων Ομοιότητας Αρχικών Ομάδων

- Για κάθε extended tree υπολογίζονται τα pq-grams trees.
- Ως pq-grams tree ορίζεται το δέντρο που περιέχει έναν κόμβο με $(p-1)$ προγόνους και q παιδιά.
- Όλα τα pq-grams ενός δέντρου αποτελούν το προφίλ του δέντρου $\mathbf{P}^{p,q}(\boldsymbol{T})$.
- Η απόσταση μεταξύ δύο δέντρων ορίζεται ως εξής:

$$\mathrm{d}^{p,q}(T_1, T_2) = 1 - 2\frac{|P^{p,q}(T_1) \cap P^{p,q}(T_2)|}{|P^{p,q}(T_1) \cup P^{p,q}(T_2)|}$$



Μεθοδολογία

Υπολογισμός Πινάκων Ομοιότητας Αρχικών Ομάδων

Λεξιλογική ομοιότητα των αλλαγών.

- Προεπεξεργασία κειμένου των αλλαγών πηγαίου κώδικα.

Βήματα Επεξεργασίας	Περιγραφή Βήματος	Κείμενο Αλλαγής
Αρχική Κατάσταση	-	“If (isSorted()) { sortDescending();}”
1ο Βήμα	Punctuation Removal	“If isSorted sortDescending”
2ο Βήμα	Tokenization	[“if”, “isSorted”, “sortDescending”]
3ο Βήμα	Split Camel Case	[“If”, “is”, “Sorted”, “sort”, “Descending”]
4ο Βήμα	Removal of Stop Words	[“Sorted”, “sort”, “Descending”]
5ο Βήμα	Lowercase	[“sorted”, ”sort” ,”descending”]

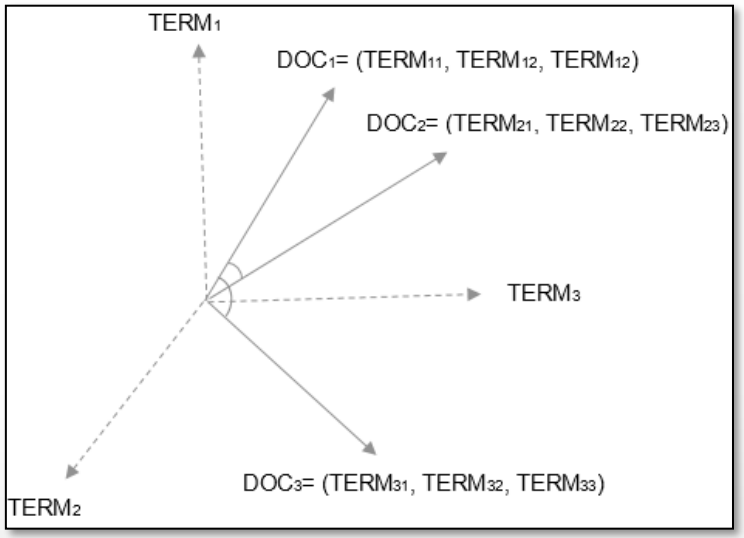
Μεθοδολογία

Υπολογισμός Πινάκων Ομοιότητας Αρχικών Ομάδων

- Διανυσματοποίηση των code changes – Vector Space Modelling.
- Χρήση του αλγορίθμου Term Frequency – Inverse Document Frequency.

$$TF.IDF(w, cc) = TF(w, cc) * IDF(w) = TF(w, cc) * \log(\frac{TC}{DF(w)})$$

  - $\boldsymbol{TF(w, cc)}$ η συνάρτηση εμφάνισης του όρου **w** στο **cc.**
  - $\boldsymbol{TC}$ ο συνολικός αριθμός των code changes
  - $\boldsymbol{DF(w)}$ το πλήθος των code changes που περιλαμβάνουν το **w**.



- Χρήση της μετρικής ομοιότητας cosine similarity:

$$\cos.\mathrm{sim}(cc_1, cc_2) = \frac{(cc_1 \cdot cc_2)}{(||cc_1|| * ||cc_2||)} = \frac{\sum_{i=1}^{N}\{tfi(w_i, cc_1) * tfi(w_i, cc_2)\}}{\sqrt{\sum_{i=1}^{N} tfi^2(w_i, cc_1)}\sqrt{\sum_{i=1}^{N} tfi^2(w_i, cc_2)}}$$

Μεθοδολογία

Τελική Ομαδοποίηση

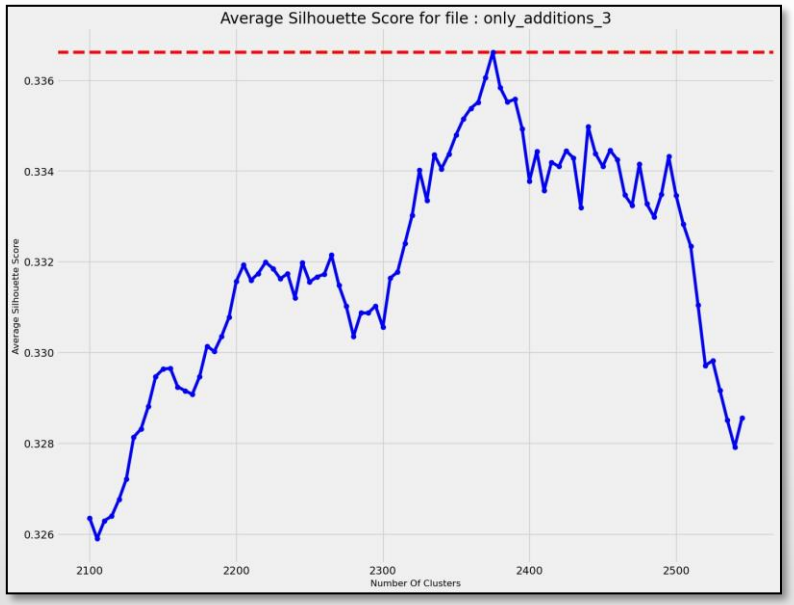
- Χρήση αλγορίθμου Agglomerative Hierarchical Clustering.
- Υπολογισμός απόστασης ομάδων με τη μέθοδο Average Linkage.

$$d(r,s) = \frac{1}{n_r \cdot n_s} \sum_{i}^{n_r} \sum_{j}^{n_s} d(r_i, s_j)$$

- Επιλογή βέλτιστου αριθμού ομάδων με τη χρήση της μετρικής Average Silhouette.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$\boldsymbol{a(i)} = \frac{1}{|C_i|-1} \sum_{\substack{j \in C_i \\ j \neq i}} d(i,j) \text{ και } \boldsymbol{b(i)} = min \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$



- Βέλτιστη τιμή αριθμού ομάδων:

$$k_{opt} = \{k : \max(\bar{s}(k))\}$$

Μεθοδολογία

Επιλογή Ομάδων & Εξαγωγή Προτύπων Αλλαγών Πηγαίου Κώδικα

- Επιλογή βέλτιστων ομάδων με τη χρήση τριών παραμέτρων:
  - Μέγεθος Ομάδας
  - Συνοχή της Ομάδας:

$$cohesion = 1 - \frac{1}{|C| - 1} \sum_{x \in C} d(x, centroid)$$

  - Αριθμός διαφορετικών αποθετηρίων
- Τα **centroids** των ομάδων αποτελούν τα τελικά πρότυπα.

$$centroid = \min\left(\frac{1}{|C| - 1}\right) \sum_{\substack{j \in C \\ j \neq i}} d(i, j)$$

Αξιολόγηση & Αποτελέσματα

Κύριος Στόχος Αξιολόγησης:

- Ανίχνευση και καταμέτρηση των προτύπων (centroids) στα Code Diffs του συνόλου αξιολόγησης.

Σύνολο Δεδομένων Αξιολόγησης:

- 300 GitHub Αποθετήρια
- **114.386** Code Diffs

Κριτήρια Ανίχνευσης:

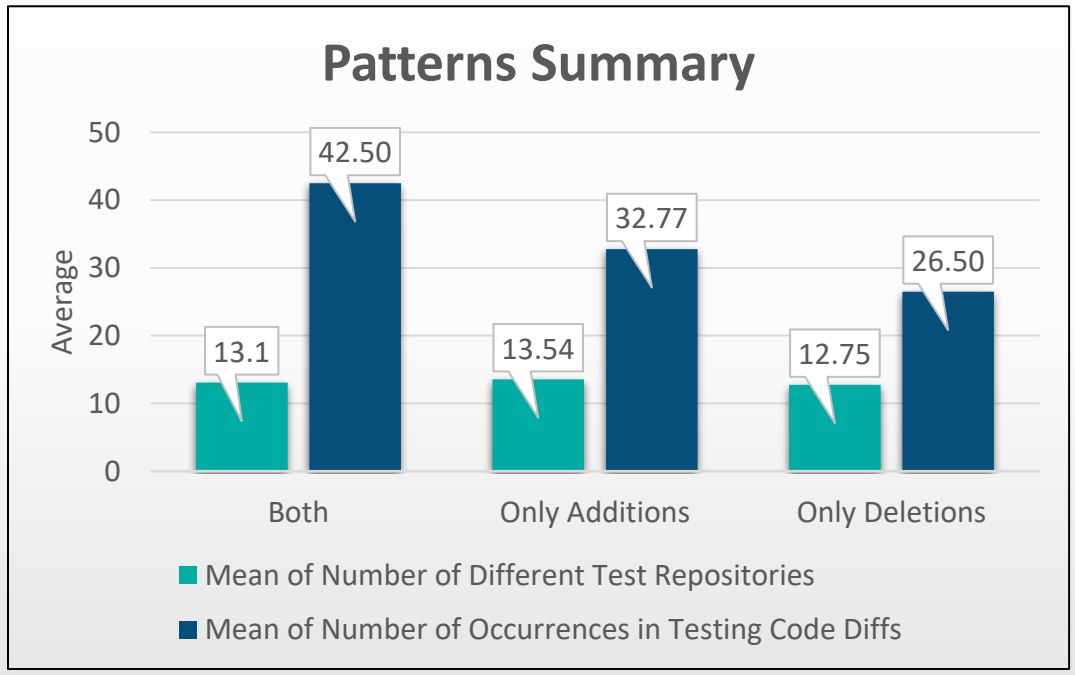
- $SequenceMatcher.ratio(sequence1, sequence2) = {}^{2.0*M}/_{T}$
  - T → Συνολικός αριθμός στοιχείων των δύο ακολουθιών.
  - M → Αριθμός των Matches.
  - Κατώτατο όριο ομοιότητας ακολουθιών: **0.8**

Πρότυπα Αλλαγών Πηγαίου Κώδικα

- Συνολικά **27** Πρότυπα Αλλαγών Πηγαίου Κώδικα

Αξιολόγηση & Αποτελέσματα

Σύνοψη Αξιολόγησης



- Μέσος όρος εμφάνισης προτύπων σε διαφορετικά αποθετήρια λογισμικού.
- Μέσος όρος εμφάνισης προτύπων στο σύνολο των αλλαγών του συνόλου δεδομένων αξιολόγησης.

Αξιολόγηση & Αποτελέσματα

Both Code Changes Patterns

Code Deletions	Code Additions	Repositories	OTCD*
public String name () {	public String getName() {	18	144
StringBuffer buffer = new StringBuffer ();	StringBuilder buffer = new StringBuilder ();	14	51
e.printStackTrace();	logger.error("", e);	6	10

out.close();	if (out!= null) {
	out.close();
	}
		27	46

*OTCD = Occurrences in Testing Code Diffs

Αξιολόγηση & Αποτελέσματα

Only Additions Code Changes Patterns

Code Additions	Repositories	OTCD

if ($Variable == null) {
  throw new NullPointerException("$Variable can’t be null");
}
 	
	6	14

@Override	30	68
this.$Variable = $Variable;	20	35
file.close();	19	32

Αξιολόγηση & Αποτελέσματα

Only Deletions Code Changes Patterns

Code Deletions	Repositories	OTCD
System.out.println();	20	50
e.printStackTrace();	20	39

} catch (Exception e) {
throw e;
}
	5	7

long time = System.currentTimeMillis();	6	10

Συμπεράσματα

Περιγραφή Προβλήματος

- Επανειλημμένη αντιμετώπιση παρόμοιων σφαλμάτων κατά την ανάπτυξη λογισμικού.

Ανάγκη Δημιουργίας Συστήματος που:

- Βοηθά στη γρήγορη και αποδοτική συγγραφή επαναχρησιμοποιήσιμου πηγαίου κώδικα.

Η Υλοποίηση του Συστήματος:

- Οδηγεί στην εύρεση συχνών αλλαγών που χρησιμοποιούνται ευρέως στα πιο δημοφιλή αποθετήρια λογισμικού και άρα μπορούν να θεωρηθούν πρότυπα.
- Βοηθά στην αποσφαλμάτωση και την βελτίωση επαναχρησιμοποιήσιμου πηγαίου κώδικα.

Μελλοντική Εργασία

Επεκτάσεις & Μελλοντική Εργασία

- Επέκταση του συνόλου δεδομένων εκπαίδευσης χρησιμοποιώντας πολλαπλές γλώσσες προγραμματισμού.
- Χρήση βάσης δεδομένων για πιο αποτελεσματική αποθήκευση και ανάκτηση της πληροφορίας.
- Χρήση της κάρτας γραφικών κατά τον υπολογισμό των πινάκων ομοιότητας.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά:

- Τον κ. Ανδρέα Συμεωνίδη
- Τον Θωμά Καρανικιώτη
- Όλους εσάς για την προσοχή σας!

Q&A