[20]

Scalabrino, S., Linares-Vásquez, M., Oliveto, R., & Poshyvanyk, D. (2018). A comprehensive model for code readability. Journal of Software: Evolution and Process, 30(6), e1958. <https://doi.org/10.1002/smr.1958>

**KEYWORDS AUTOR:**

code readability, quality warning prediction, textual analysis

**TÍTULO**:

Un modelo integral para la legibilidad del código.

**PARA ANEXAR A DOCUMENTO:**

**\*\* INICIO \*\*\***

**DATASET y HERRAMIENTA**:

<https://dibt.unimol.it/report/readability/>

**LENGUAJE DE PROGRAMACIÓN:**

Java

**TAGS**

LEGIBILIDAD; MODELOS; CARACTERISTICAS; MACHINE LEARNING; JAVA; EVOLUCION;

Nota: tener en cuenta lo que no está en rojo (es info adicional, por si se necesita luego).

**\*\* FIN \*\*\***

**RESUMEN COMPLETO**

otras notas:

|  |  |  |
| --- | --- | --- |
| (26)  NOO | Binkley et al | Propuso la herramienta QALP para calcular la similaridad textual entre el código y sus comentarios asociados.  La baja consistencia entre identificadores y comentarios en un componente de software es altamente propenso a fallas. |

3 modelos para la predicción de la legibilidad han sido propuestos y realizan el siguiente proceso:

1. Medir aspectos específicos del código fuente.
2. Usar las métricas propuestas para entrenar un clasificador binario que sea capaz de decir si un fragmento de código es o no legible.

El estado del arte define más de 80 características, divididas en 2 categorías: características estructurales y visuales. Proponen un nueva categoría de características textuales, las cuales capturan la semántica del dominio y adiciona una nueva capa de información semántica del código.

detalles del experimento;

más de 600 fragmentos de código.

más de 5000 participantes.

EVOLUCIÓN DE LAS CARACTERÍSTICAS DE LA LEGIBILIDAD

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | AUTOR | MODELO | Resultado |
| 2010 | Buse y Weimer | Primer modelo de legibilidad del software, evidenciando que un aspecto subjetivo como la legibilidad puede ser capturado y predecir de forma automática.  modelo opera como un clasificador binario, entrenado con fragmentos de código etiquetados manualmente.  120 participantes; 100 pequeños fragmentos; 12000 juicios humanos.  Consideran sólo el aspecto **estructural** | Precisión 80% |
|  | Posnett et al | Modelo simple de legibilidad, con menos características que el de Buse et al: líneas de código, entropía y métricas de volumen de Halstead |  |
|  | Dorn | Modelo generalizado con características organizadas en 4 categorías: **visuales, espaciales, alineación y lingüísticas**. |  |
|  | Scalabrino et al | Proponen una nueva categoría: Textual, con 7 características, las cuales capturan la semántica del dominio y adiciona una nueva capa de información semántica del código. Realiza un análisis sintáctico de los términos (palabras del código), a los cuales se les hace un pre procesamiento. |  |

Pre procesamiento de los términos en el modelo de Scalabrino et al:

1. Eliminar tokens no textuales, como operadores, símbolos especiales y palabras clave del de lenguaje de programación;

2. Dividir las fichas restantes en palabras separadas usando la puntuación inferior o los separadores de mayúsculas y minúsculas; por ejemplo, getText se divide en get y text;

3. Eliminar las palabras que pertenecen a una lista de palabras vacías (por ejemplo, artículos, adverbios).

4. Extraer los tallos de las palabras usando el algoritmo de Porter.

\*\*\* es alta correlación, \*\* es media y \* es baja

CARACTERÍSTICAS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Categoría | características | Buse y Weimer (2010) [10][20] | Posnett et al (2011) [12][20] | Dorn (2012) [13][20] | Scalabrino et al. (2016-2018) [11][20] |
| Estructural | Longitud promedio de línea (caracteres) | **\*\*\*** |  |  |  |
| Estructural | Longitud máxima de línea (caracteres) | **\*\*\*** |  |  |  |
| Estructural | Cantidad promedio de identificadores | **\*\*\*** |  |  |  |
| Estructural | Cantidad máxima de identificadores | **\*\*** |  |  |  |
| Estructural | Identación promedio (espacio en blanco anterior) | **\*\*** |  |  |  |
| Estructural | Identación máxima (espacio en blanco anterior) | **\*\*** |  |  |  |
| Estructural | Cantidad promedio de palabras clave | **\*\*** |  |  |  |
| Estructural | Cantidad máxima de palabras clave | **\*** |  |  |  |
| Estructural | Longitud promedio de Identificadores | **\*** |  |  |  |
| Estructural | Longitud máxima de Identificadores | **\*\*** |  |  |  |
| Estructural | Cantidad promedio de números | **\*** |  |  |  |
| Estructural | Cantidad máxima de números | **\*** |  |  |  |
| Estructural | Cantidad promedio de paréntesis | **\*\*\*** |  |  |  |
| Estructural | Cantidad promedio de períodos | **\*\*\*** |  |  |  |
| Estructural | Cantidad promedio de líneas en blanco | **\*\*** |  |  |  |
| Estructural | Cantidad promedio de comentarios | **\*\*** |  |  |  |
| Estructural | Cantidad promedio de comas | **\*\*** |  |  |  |
| Estructural | Cantidad promedio de espacios | **\*** |  |  |  |
| Estructural | Cantidad promedio de asignaciones | **\*** |  |  |  |
| Estructural | Cantidad promedio de bifurcaciones (if) | **\*** |  |  |  |
| Estructural | Cantidad promedio de bucles (for, while) | **\*** |  |  |  |
| Estructural | Cantidad promedio de operadores aritméticos | **\*** |  |  |  |
| Estructural | Cantidad promedio de operadores de comparación | **\*** |  |  |  |
| Estructural | Cantidad máxima de ocurrencias de algún carácter | **\*\*** |  |  |  |
| Estructural | Cantidad máxima de ocurrencias de algún identificador (fin) | **\*\*** |  |  |  |
| Estructural | Líneas de código |  | **\*\*** |  |  |
| Estructural | Entropía |  | **\*** |  |  |
| Estructural | Métrica de volumen de Halstead |  | **\*** |  |  |
| Visual | Longitud de línea |  |  | **x** |  |
| Visual | Longitud de indentación |  |  | **x** |  |
| Visual | Asignaciones |  |  | **x** |  |
| Visual | comas |  |  | **x** |  |
| Visual | Comparaciones |  |  | **x** |  |
| Visual | Ciclos |  |  | **x** |  |
| Visual | Paréntesis |  |  | **x** |  |
| Visual | Periodos |  |  | **x** |  |
| Visual | Espacios |  |  | **x** |  |
| Visual, Espacial | Comentarios |  |  | **x** |  |
| Visual, Espacial | Palabras clave |  |  | **x** |  |
| Visual, Espacial, Textual | Identificadores |  |  | **x** |  |
| Visual, Espacial | Números |  |  | **x** |  |
| Visual, Espacial, Alineación | Operadores |  |  | **x** |  |
| Espacial | Cadena de palabras Strings |  |  | **x** |  |
| Espacial | Literales |  |  | **x** |  |
| Alineación | Expresiones |  |  | **x** |  |
| Textuales | Consistencia de los comentarios e identificadores - CIC |  |  |  | **\*** |
| Textuales | Términos de identificadores en diccionario - ITID |  |  |  | **\*** |
| Textuales | Identificadores de significado restringido - NMI |  |  |  | **\*** |
| Textuales | Legibilidad de los comentarios - CR |  |  |  | **\*\*** |
| Textuales | Número de significados - NM: Cuántos significados tiene una misma palabra. |  |  |  | **\*** |
| Textuales | Coherencia textual - TC: Mide la cohesión de las clases dentro del código. |  |  |  | **\*** |
| Textuales | Número de conceptos - NOC |  |  |  | **\*** |

**CARACTERÍSTICAS DE SCALABRINO ET AL**

**1) Consistencia de los comentarios e identificadores**

CIC (m) = |comentarios(m) <intersección> identificadores(m) |/ |comentarios(m) <union> identificadores(m)|

donde,

m= método

Teniendo en cuenta que pueden existir sinónimos de las palabras en comentarios y en identificadores, se tiene una variación de la ecuación:

CIC(m)sin = |comentarios(m) <interseccion> identificadores(m) <union> sinónimos(m)|/ |comentarios(m) <union> identificadores(m) <union> Sinónimos(m)|

donde,

sinónimos, es el conjunto de todos los sinónimos de los términos que están en los identificadores.

**2) Términos del identificador en diccionario**

está inspirada por las características de lenguaje natural introducidas por Dorn.

ITID(I) = |Term(I)<intersección>Diccionario| / |Terminos(I) |

donde,

Términos = conjunto de términos extraídos de una línea I de un método.

Diccionario = Es un conjunto de palabras en un diccionario

**3) Identificadores de significado estrecho**

Los términos deben tener un significado específico, no genérico que pueda generar confusiones.

NMI(I) = Sumatoria (particularidad(t)) de todo t que pertenece a I

donde,

t= es un término extraído de la línea de código I.

particularidad(t) =es calculado como el número de saltos desde el nodo t que lo contiene hasta el nodo raíz del árbol hypernym de t. Usan el árbol hypernym/hyponym del lenguaje inglés definido para WordNet.

**4) Legibilidad de los comentarios**

Calcula la legibilidad de los comentarios (CR) usando el índice Flesh-Kincaid comúnmente utilizado para evaluar la legibilidad de textos en lenguaje natural. El índice considera 3 tipos de elementos: palabras, sílabas y frases.

FK(S) = 206.835 - 1.015\*(palabras(S)/frases(S)) - 84.600\*(sílabas(S)/Palabras(S))

donde,

S= es el fragmento de código fuente.

Se calcula el CR al (1) juntar todas las líneas comentadas del fragmento S; (2) unir los comentarios con un carácter ".", Para asegurarse de que no se unan diferentes comentarios creando una sola frase; y (3) calcular el índice de Flesch-Kincaid en dicho texto.

**5) Número de significados**

El significado de una palabra puede ser polisémico (tener más de un significados). La característica mide el número de significados (NM) o el nivel o polisemia de un fragmento. Miden el número de significados derivados de WordNet.

**6) Coherencia textual**

Cuando un fragmento tiene baja cohesión es más difícil de comprender. La coherencia textual de un fragmento puede ser usada para estimar el número de conceptos implementados por un fragmento de código fuente.

Se analiza el código fuente y se construye el árbol de sintaxis abstracta (AST) para detectar bloques sintácticos. Se calcula la superposición de vocabulario entre todos los posibles pares de bloques sintácticos distintos. La coherencia textual (TC) de un fragmento se puede calcular como el máximo, el mínimo o la superposición promedio entre cada par de bloques sintácticos.

**7) número de conceptos**

Captura directamente el número de conceptos implementados en un fragmento de código a nivel de línea. Crean un documentos por cada línea. Utiliza un técnica de clustering basada en densidad, DBSCAN.

Miden la distancia entre 2 documentos así:

NOC dista (d1,d2) = |d1 <interseccion> d2| / |d1 <union> d2|

número de conceptos de un fragmento m:

NOC(m) = |Cluster(m)|

normalizando NOC:

NOC norm(m) = |Cluster(m)| / |Documentos(m)|

**8) Legibilidad Vs Comprensibilidad**

Legibilidad mide el esfuerzo de el desarrollador para acceder a la información contenida en el código, mientras la comprensibilidad mide la complejidad de dicha información.

**FRAGMENTOS DE CÓDIGO USADOS EN LOS MODELOS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Autor | Lenguaje de programación | Cantidad | Descripción | Participantes | Anotados |
| Buse y Weimer (2010) | Java | 100 | Tamaño de cada fragmento: 7 LOC  Problema: incompletos | Total: 120  Tipo: estudiantes | 12000 |
| Dorn | CUDA, Java y Python | 360 (120 de cada lenguaje) | Tamaño variado: entre 10 a 50 LOC  Problema: incompletos | Total: 5468  Tipo: estudiantes y profesionales | 76741 |
| Scalabrino et al | Java | 200 | Tamaño: entre 10 a 50 LOC  Métodos completos de los proyectos de código abierto: jUnit, Hibernate, jFreeChart, y ArgoUML | Total: 9  Tipo:estudiantes | 1800 |

**Modelos de legibilidad de software**

|  |  |  |
| --- | --- | --- |
| Modelo | Descripción | Resultado promedio de precisión %  Dbw | Ddorn | Dscal | Gwm | Gam |
| Buse y weimer (2010) | Características estructurales  Más relevantes:  Número máximo de identificadores, número promedio de identificadores y máxima longitud de línea.  Número promedio de comentario, número máximo de identificadores y número promedio de operadores.  Longitud promedio de indentación, longitud máxima de identificadores y número promedio de comentarios. | 81 | 78.6 | 72.5 | 77.1 | 77.4 |
| Posnett et al | Características estructurales  3 más relevantes:  Volumen, entropía, líneas. | 78 | 72.8 | 66 | 71.5 | 72.3 |
| Dorn | Características estructurales y visuales  3 más relevantes:  Área de strings/comentarios\*, área de operadores/comentarios\* y área de identificadores/comentarios\*.  número de comentario (visual Y), número de numeros (visual Y) y número de comentarios (visual X).  Condicionales (DFT) | 81 | 80 | 75.5 | 78.8 | 78.8 |
| Scalabrino (características textuales) | Características textuales:  CR\*, TC promedio, NOC.  ITID promedio, TC máximo, TC mínimo.  Métodos:  Regresión logística para la clasificación binaria.  Para evaluarlo: aprendizaje de máquina: red de Bayes\*, RandomForest\*, ML perceptron y SMO. Usaron las métricas precisión (accuracy) y AUC. | 74 | 78.1 | 76.5 | 77 | 76.2 |
| Modelo con características estructurales y visuales | Características de Buse, Posnett y Dorn.  Métodos:  Regresión logística para la clasificación binaria.  Para evaluarlo: aprendizaje de máquina: red de Bayes\*, RandomForest\*, ML perceptron y SMO. Usaron las métricas precisión (accuracy) y AUC. | 83 | 80.6 | 77 | 79.9 | 80.2 |
| Combinado: modelo integral de legibilidad del código | Todas las características estructurales, visuales y textuales de los modelos anteriores.  Métodos:  Regresión logística para la clasificación binaria.  Para evaluarlo: aprendizaje de máquina: red de Bayes\*, RandomForest\*, ML perceptron y SMO. Usaron las métricas precisión (accuracy) y AUC. | 87 | 83.9 | h84 | 84.4 | 85 |

Nota:

General wm: se calcula como la media ponderada de los valores de precisión para cada conjunto de datos, donde los pesos son el número de fragmentos en cada conjunto de datos.

General am: se calcula como la media aritmética de los valores de precisión para cada conjunto de datos.

Para identificar los atributos más relevantes, usaron el algoritmo de selección de atributos ReliefF, obteniendo las 3 más importantes por cada modelo en cada conjunto de datos.

**Método usado:**

Regresión logística, porque se ha demostrado que es muy eficaz en la clasificación binaria.

Para evitar el sobreentrenamiento, realizaron la selección de características mediante la selección lineal hacia adelante con una estrategia de envoltura disponible en el Caja de herramientas de aprendizaje automático de Weka. En la estrategia de selección de envoltura, cada subconjunto de características candidato se evalúa a través de la precisión del clasificador entrenado y probado utilizando solo dichas características. El resultado final es el subconjunto de características que obtuvieron la máxima precisión.

Para evaluar la significancia de los resultados, usaron la prueba estadística de diez veces (tenfold) la validación cruzada y realizaron análisis estadístico con la prueba Wilcoxon, efecto de tamaño, e intervalo de confianza, para medir la significancia de las diferencias entre los resultados de precisión de los modelos.

Para evaluar los modelos de legibilidad, usaron técnicas de aprendizaje de máquina cómo red de Bayes\*, RandomForest\*, ML perceptron y SMO. Usaron precisión (accuracy) y AUC logrado cuando usaron la regresión logística.

La mejora en la precisión de la predicción de la legibilidad da como resultado una mejor predicción de las advertencias de FindBugs, demostrando que un código poco legible es más propenso a tener problemas.