Comment-Grade: Framework for Comment Quality Assessment in C / C++ based on Industry Practices

Table 1: Features – semantics and structure

SL No.	Feature	Algorithm
1	Semantic	Vector representation (Software Development trained corpus) - For each word in comments belonging to POS tags [(NN)*, (VB)*, (JJ)*]. Dimensions reduced to 200 for every comment, during training using a 2 dense layer LSTM (hidden layer size, activation: 200, LeakyRelu:64:tanh) and 1 fully connected Output layer (3, softmax)
2	Syntax	*Count* of words having POS tags [NNP, NNPS and SYM] and POS tags [(NN)*, (VB)*, (JJ)*, (RB)*] - [NNP, NNPS]
3	Number of Comments Tokens	Normalised count of comment tokens
4	Description	Algorithm to understand the structure of the Stanford dependencies and their values – $nsubj,\ vmod,\ conj\_and,\ root$
5	Operational	Algorithm to understand the structure of the Stanford dependencies and their values – $nsubj,\ mark,\ in$
6	Scope Score (Empirically arrived)	$\frac{1}{1+log(\sum_{n=1}^{Id} n*distance)}$, Id is the number of constructs in scope, $distance$ is the line distance from comment

Table 2: Features – Knowledge Domains of relevant comment categories - PART I

#	Feature	Extraction Logic
1	No. of SD Concepts	*Count* of keyword matches with `SD Ontology`
2	Mapping to AD	*Count* of keyword matches with the enumerated concepts
3	Developer Details	*Count* of matches with developer names using NER
4	Description of Dataset	I. *Count* of keyword matches (**syntax, semantic**) in comment text with the following – a) Instances part of class ‘Operations as part of Algorithms’, ‘Operations as part of Data structure’ (enumerated in `SD Ontology`) b) Data type and alloc keywords such as - [“string”, “list”, “array”, “matrix”, “memory”, “alloc”, “malloc”, “static”, “calloc”, “dynamic”, “pointer”, “binary”, “hex”, “logs”, “buffer”, “static”, “space”, “disk”] c) Units and dimensions - [“size”, “shape”, “dimension”, “byte”, “kilo”, “mega”, “giga”, “tera”, “kb”, “mb”, “gb”, “tb”] d) ‘N * N’ kind of keyword matches, specifically using the regex – $[0-9a-zA-Z]?[]?[0-9a-zA-Z]$
5	Working Summary – Interaction	I. *Count* of keyword matches in comment text with the following – a) Instances part of class ‘Divide and Conquer/ Greedy Algorithms’, ‘Sorting/ Searching Algorithms’, ‘Dynamic Programming’ (enumerated in Software Domain Ontology, ‘Operations as part of Algorithms’, ‘Operations as part of Data structure’ (`SD Ontology`) b) Doxygen keywords such as - [“param”, “return”, “arg”, “class”, “parblock”, “throw”]. If the number of matches is greater than 4, then its weight is doubled II. *Count* of Verb (‘VB*’) tokens present in the list of POS tags
6	Working Summary – Design	I. *Count* of keyword matches in comment text with the following – a) Instances part of class ‘Divide and Conquer/ Greedy Algorithms’, ‘Sorting/ Searching Algorithms’, ‘Dynamic Programming’ ‘Operations as part of Algorithms’, ‘Operations as part of Data structure’, ‘Properties of Datastructure / Function / Blocks’, ‘Data-Structure and its Components’, ‘Time Complexity / Space Complexity’, ‘Memory operations’, ‘Exceptions’, ‘Threads’ (`SD Ontology`, Example in **??**)
Final Score calculation – determined after empirical analysis		
Keyword Matches signifies syntactic matches (through exact string match, stemmed or lemmatised match), top 10 similar words or through cross similarity match ($>$ than a threshold of 0.57 based on empirical analysis)) using pretrained embeddings `SD2Vec` **Scores normalised using mean (μ) and standard deviation (σ). The formula used is $(datapoint - \mu)/\sigma$, It is then transformed to a range of [-1,1] using a hyperbolic tangent function $tanh$ or into the range [0,1] using $sigmoid$**		

Table 3: Features – Knowledge Domains of relevant comment categories - PART II

#	Feature	Extraction Logic
7	Exception, Memory Related	I. *Count* of matches in comment text with the following – a) Instances part of class ‘Time Complexity / Space Complexity / Memory / Exception’ (enumerated in `SD Ontology`) b) Matches with Exception list (Java and C / C++ errors)
8	Libraries / Imports	I. *Presence* of an import statement nearby (comment - identifier distance < 3 lines, 8 columns) – II. *Count* of keyword matches with .h
9	Build Instructions	I. *Count* of matches in comment text with the following - Build keywords - [“gcc”, “g++”, “make”, “config”, “build”, “install”, “mkdir”, “cd”, “cmake”, “–”, “git”, “.tar”, “.gz”, “.zip”, “cxx”, “clang”,“.dll”]
10	Project Management	I. *Count* of matches in comment text with the following - – a) Keywords such as - [“issue”, “commit”, “svn”, “bug”, “jira”, “git”] b) Regular Expression for bug id (based on observation of format in BugZilla, RationalRose, etc.) $(\#[0-9a-f]+)\vert((([0-9a-zA-Z]+:)+[0-9a-zA-Z]+)\vert((([0-9].)+[0-9])$
Final Score calculation – determined after empirical analysis		
Matches signifies syntactic matches (through exact string match, stemmed or lemmatised match), top 10 similar words or through cross similarity match ($>$ than a threshold of 0.57 based on empirical analysis)) using pretrained embeddings `SD2Vec` **Scores normalised using mean (μ) and standard deviation (σ). The formula used is $(datapoint - \mu)/\sigma$, It is then transformed to a range of [-1,1] using a hyperbolic tangent function $tanh$ or into the range [0,1] using $sigmoid$**		

Table 4: Features – code-comment correlation

SL No.	Feature	Extraction Logic
1	AST symbols	*Count* of keyword matches in comment text with AST symbols extracted from source files
2	Comment Placements	Measured by the type of constructs present at the nearest distance to deduce - Inline, Global or Block level
3	Scope Score	$\frac{1}{1+log(\sum_{n=1}^{Id} n*distance)}$, Id is the number of constructs in scope, $distance$ is the line distance from comment
4	Program Domain Identifier Matches	*Count* of syntactic and semantic (cosine similarity) matches of Program Domain concepts identified in comments with tokenised identifiers which are part of the comment scope
5	Problem Domain Identifier Matches	*Count* of syntactic and semantic (cosine similarity) match of Problem Domain concepts identified in comments with tokenised identifiers which are part of the scope of the comment
6	Structure Matches	*Count* of matches with data types, type of AST node and type of operators

Count is not a direct one, it is a score based algorithm based on the cosine similarity of the concepts extracted. Highly unrelated concepts can signify inconsistency and the vice versa can indicate redundancy.
The thresholds of the cosine distances are determined based on empirical analysis