THE UNIVERSITY OF AUCKLAND

Department of Electrical and Computer Engineering

SOFTENG 788 – PROJECT Y REPORT

2018

Clustering Source Code Comments

Written and Submitted By:

Welisarage Sudara Fernando

Student ID: 388335035

Supervised By: Dr. Kelly Blincoe

DECLARATION OF ORIGINALITY

I hereby declare that I am the author of this report and the content of the report is not copied from or is not submitted previously to any other qualification.

Submitted by : Welisarage Sudara Fernando

Clustering Source Code Comments

Welisarage Sudara Fernando
Dept. of Electrical and Computer Engineering
University of Auckland
Auckland, NZ 1010
wfer594@aucklanduni.ac.nz

***Abstract*—Text clustering and classification techniques are used in characterising text documents into one or more categories based on various factors. This study focuses on classifying source code comments into a set of categories by utilising several natural language processing techniques such as TF-IDF vectorising, stop words elimination and frequency distribution count of token in the text document. Furthermore, the comments to perform classification are also collected by implementing a source code scanner for Java.**

***Index Terms*—text classification, comments, tf-idf vectorising, dbscan, k-means**

I. Introduction

Data classification highlights the relationships between data points. It uncovers information otherwise hidden in plain-sight which can be visualised and processed to make better-informed decisions. Text clustering is a form of data classification which divides a set of text documents into one or more categories based on their underlying linguistic content. It can be visualised as a mapping function which maps the given documents into categories. It can be one-to-one as well as one-to-many [1]. The semantic factors that influence text classification results are discussed, focusing on the ones that apply on "short text"s, such as source code comments which are usually in the range of 100 to 150 words.

Natural Language Processing (NLP) has been a vital topic in computer science. It involves analysing human languages and recognising the connotative idea of text documents. NLP is a subfield of Artifical Intelligence (AI) and Machine Learning (ML) which extends a computer program's understanding of human languages. It powers several linguistic tasks such as text summarization, classification, translation and manipulation. Human languages are inherently complex. Some complexities of the English language are different literal meaning from the actual implied meaning (idioms and slang), different meaning of the same word depending on the context and different style of writing (American and British). However, advanced NLP techniques have progressively enabled the understanding of the intricate languages [2] which innately enables excellent text manipulation using computer systems. This paper explores different NLP techniques and algorithms which can be utilized for meaningful text classification.

The clustering algorithms discussed in this research are based on *unsupervised learning* methods which utilises a set of features F_1,F_2,F_3...F_n obtained from n observations to produce a meaningful visualization of data. This is opposed to *supervised learning* which predicts a response R over the feature set F_1,F_2,F_3...F_n using a set of f features $\{F_1,F_2,F_3...F_n\}$ measured from n observations and a response relation R_R which is also measured on the same n observations [3].

The only input for an *unsupervised learning* approach is the dataset. The dataset for this project is obtained through a source code scanning program designed to scan through Java source code. An *unsupervised learning* procedure does not require a pre-determined response relation (a model) since the process does not involve in any prediction. In a *supervised learning* environment, the results can be cross-validated with the predetermined model. However, in an *unsupervised* approach, there is no universally accepted mechanism to validate the results since the problem itself was unsupervised without an initial model [3]. The algorithms used in this research visualise the results, but unable to cross-validate due to the unsupervised nature of the problem.

Two main outcomes of this project are retrieving comments from source code files and classifying them into several categories.

A. Retrieving source code comments

The dataset was obtained through a source code scanner program which was designed to scan through a given set of Java source code files collecting both single-line as well as multi-line comments. More than 5,000 Java source files were obtained from several well-established and maintained open source projects such as Firebase Admin API (Java), Android Open Source Project and Apache Hadoop. The comments scrapped from the used files were written in English which adheres to correct English language grammar and semantics.

B. Comment clustering

The collected comment data is then processed using various NLP techniques and data cleansing methods. After cleansing it is represented it as n x n matrix. The matrix is then vectorised using a TF-IDF Vectoriser and then fed into the algorithm. The DBSCAN clustering algorithm was used as the clustering algorithm for this project. Finally, the result set is then examined to identify and visualise the cluster output.

II. Literature Review

A. Retrieving source code comments

Source code parsing and lexical analysis of the programming languages are used in many software products such as

Integrated Development Environments (IDE) and other text editors to provide visual feedback of the code that a developer is working on. These tools utilise regular expressions and other language-specific tools to construct a data structure called *abstract syntax tree* (AST) which is composed of a sequence of characters [4]. Several authors have attempted to design software tools to obtain an AST from Java source code.

One such implementation is to obtain an XML view of the Java source code, named JavaML [4]. The authors formalise a *document type definition* (DTD) which describes the conversion of Java grammar into XML nodes. Then a computer program based on IBM Jikes Java compiler framework [5] was used to parse the standard Java source files and convert them into JavaML documents. These JavaML documents can be utilised by other programs which support XML file parsing. Java is a live language which gets continuously updated. Hence it is vital to keep the language parsers up to date which is also being discussed by the authors in their paper. However, the proposed algorithm only supports Java 5.0 and is unable to process the new Java lexical structure introduced in newer Java versions. Also, the suggested procedure generates a complex and bulky XML document even for simpler Java source code. XML itself is verbose and converting Java source with all its individual properties result in a massive XML document which makes it challenging to cross-validate.

Contrary to AST, Token-based source code feature identification is also found in the literature. The token-based approach is linear complexity in both time and space and does not require the whole document to be parsed at once [6]. It even works when the source code is syntactically incorrect or incomplete. It is more suitable to identify a particular language feature (such as individual methods, variables or comments). AST-based procedures usually traverse the whole document several times to classify the distinct language components whereas the token-based approaches tokenise the source code to recognise the required language feature.

An open-source library named *"javalang"* provides a lexer and a parser for Java 8.0 syntax. *"javalang"* uses token-based feature extraction to scan and tokenise a Java source code and represents it in a tree data structure [7]. It also provides a Python API to access the properties and methods listed in the Java source code. However, one shortcoming of this library was that it was unable to recognise multi-line comments.

B. Comment clustering

Several text clustering algorithms such as K-Means, DBSCAN and FTSC exist in the current literature. These algorithms are mainly categorised into prototype-based, density-based and hierarchical clustering. Prototype-based algorithms such as K-Means assigns each document into its nearest initial prototype cluster. Density-based clustering algorithms such as DBSCAN, separate documents into subsets of similar densities [8]. These algorithms differ from each other based on several factors such as scalability, use case and geometry. Each algorithm also requires a different set of parameters as inputs.

The frequent terms in text clustering is a common factor analysed by several algorithms. Frequent term Set-based clustering (FTSC) was introduced by X Liu et al. which uses frequent term sets for text clustering [9]. FTSC uses a part-of-speech tagging system to segment the terms of a text document and use a discrimination value (DV) of feature terms to find words which can reflect the caption of a text. The DV value represents a word token's ability to distinguish unique document with different contents when it is used to identify the content. Furthermore, Frequent Term Set-based Hierarchical Clustering algorithm (FTSHC) modelled after FTSC can produce overlapping clustering results which increase its reliability and accuracy. Authors claim that it is both faster and reliable than the K-Means algorithm.

FTSC improves the efficiency of clustering by choosing a set of frequent terms based on its appearance frequency as the candidate set without immediate clustering of the documents with high dimensions. Mining of the frequent terms is a two-step process where all the frequent items found in the text are filtered according to a predetermined threshold (internally known as *min_sup*) [9]. Then the general association rules are obtained for each filtered item set adhering to the rules defined in the Apriori algorithm. Each frequent term in frequent term set $\{F\}$ forms a clustering category where each category $\{f_i\}$ has the least mutual overlap [9]. It shows that there is no much difference in the clustering results obtained from FTSC and K-Means algorithms according to the quantitative analysis conducted by the authors using F-Measure on two datasets (namely *NTCIR-3* and *Reuters-21578*). However, due to the nature of the frequent term selection algorithm used in mining the datasets, FTSC is more optimised for text documents with longer and larger text data [9].

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm which is heavily adopted for text clustering. It requires two main parameters; a point neighbourhood and the minimum number of points (*min_sample*) in the neighbourhood. The cohesion of the result depends on this minimum point value [8]. The algorithm starts at an arbitrary position and if *min_sample* $\leq$ number of points in the initial position, then a cluster is formed. Otherwise, the point will be marked as noise and will select another point to repeat the process. DBSCAN produces clusters of arbitrary shape and is more sensitive for datasets with noise.

The selection of the *min_sample* and the minimum neighbourhood value (*eps*) depends on the dataset used. Setting the *min_sample* to 1 does not make any sense since every document in the corpus will then be considered as a cluster by itself. It is usually recommended to set this value to a minimum 3. But it may be possible to set a generous amount for a bigger dataset or datasets with more noise and duplicates. The eps value affects the number of clusters produced in the result (discussed in SECTION V). If the chosen amount is too small, a large part of the data will be discarded as noise. If it is too big, then the clusters will merge affecting the accuracy of the outcome.

III. Dataset preparation

The primary goal of this research is to mine comment data from a given set of Java source files and to cluster them based on their linguistic content. A Java source code scanning program was developed in Python to scan through source files and collect both single-line as well as multi-line comments. The program uses a token-based source code scanning approach since it only targets a specific language component (i.e comments). AST traverses the source file several times to identify all the language components which is an overkill for this research. Token-based approach also dramatically reduces the time and memory requirements of the scanning program [6] which is essential because the program has to scan an adequate amount of source files to obtain comment data for a meaningful clustered result set.

A. Comment data collection

The program scans the source files and identifies the comment start and end tokens depending on the comment type. It then starts capturing the comment as soon as it detects the start character and ends the comment when it meets the end character. The start character may appear in any arbitrary location in the scanned line. In case of the multiline comments, the star (*) character that appears in between the lines are omitted. To minimise unwanted omissions, it gets excluded only if it is followed by a line-end character.

TABLE I
START AND END TOKENS FOR JAVA COMMENTS

Comment Type	Start Character	End Character
Single-line	//	Line end charactor
Multi-line	/*	*/

The collected comment data is then stored in a Python dictionary along with a reference to the file.

B. Data cleaning

Data cleaning is an essential step in text clustering process. Text clustering, specially density-based clustering algorithms cluster documents depending on the neighbourhood spacial size. Several common English language terms such as articles, pronouns, prepositions and conjunctions can skew the results as they commonly occur in text documents. Also in most cases, the less frequent words in a sentence convey the meaning rather than the most occurring articles and pronouns.

1) Filtering stop words: Stop words with a high count frequency occur in a variety of languages and texts. These words can skew the frequency counting and density based classifications. They appear in several places of the sentence yet fail to convey the meaning without other content words accurately. Examples for stop words in the English language include ”of”, ”is”, ”the” etc. There are two possible ways to filter the stop words. One is to use a precompiled stop words dictionary and to refine the text document excluding the words found in the dictionary. The other option is to use a word

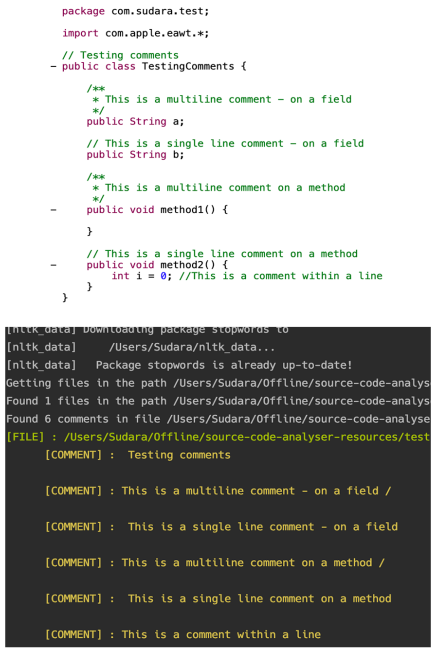

Fig. 1. Comment scanning example

frequency threshold where words with higher frequencies are considered stop words [10].

The NLTK Python package, which is used for the clustering process uses a precompiled stop words dictionary. Once the *”stopwords”* package is downloaded, the words can be filtered by specifying the language.

2) Stemming and lemmatization: Both stemming and lemmatisation are used in English language text processing. Reducing a word by analysing its prefix or suffix into its stem or root is known as stemming. Words with the same meaning with different prefixes and suffixes can affect count frequency. For example, words *”running”*, *”ran”* and *”runs”* represent the root word *”run”*. These words would be counted as different unless they were stemmed into its roots. The information retrieval of the English documents is improved by 6% - 8% after using a stemming process [11].

EnglishStemmer from *nltk.stem.snowball* package was used in the project for stemming the words into its root representation.

IV. Document feature extraction

After obtaining the dataset, the next step is to extract several document features. In this step, the raw text data will be transformed into feature vectors to be analysed by the clustering algorithm.

A. Bag of words representation

The raw corpus data cannot be fed directly into most of the algorithms for analysis. It will have to be converted into fixed size numerical feature vectors for it to be processed and analysed. This conversion of raw text data into a collection

of numerical feature vectors is known as ”Bag of Words” (”Bag of n-grams”) representation. The project applies a three-step conversion function. Firstly the words are tokenised into several token terms. Each of the tokens is given an integer id for later identification. Secondly, each token occurrence is counted and stored in a matrix (3D array). Finally, the counted tokens are weighted using TF-IDF as described below. The final matrix represents the input corpus as a set of numerical vectors. This matrix representation is then used by the DBSCAN algorithm to analyse and form the clusters. [12]

B. Count vectors

The count vector represents the frequency of a token occurrence in the given document. It is internally depicted as a matrix notation. In the matrix, a column is represented by a word or a term in the corpus and a row is represented by an individual document in the corpus. By examining each cell, we can identify the frequency count of a given word in the given text document.

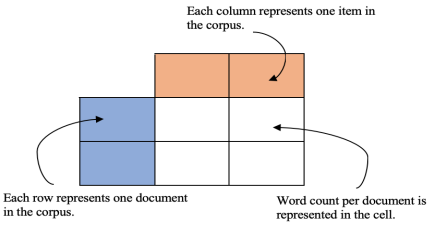

Fig. 2. Count vector representation in NLTK

The *CountVectorizer* class in the *sklearn* package was used to obtain the count vector of the collected dataset.

```
CountVectorizer Array:
[[0 1 0 0 0 0 0 0 0 1 0 0]
 [1 0 1 1 0 0 1 1 0 0 1 0]
 [1 0 1 1 1 0 0 1 1 0 1 0]
 [1 0 0 1 0 1 1 1 0 0 1 0]
 [1 0 0 1 1 1 0 1 1 0 1 0]
 [1 0 0 1 1 0 0 0 0 0 1 1]]
CountVectorizer vocabulary:
{'testing': 9, 'comments': 1, 'this': 10,
'is': 3, 'multiline': 6, 'comment': 0, 'on
': 7, 'field': 2, 'single': 8, 'line': 4,
'method': 5, 'within': 11}
```

Fig. 3. Count vector representation in NLTK

C. TF-IDF vectors

The TF-IDF (term-frequency times inverse document-frequency) represents the weight of a term in the entire document. TF-IDF weighting is essential because some scarce words in the language provide more meaning in the sentence than the other commonly occurring words. This aspect is ignored if we only count the occurrence frequency of a term rather than estimating the weighted value for contributing words.

TF-IDF is a two-step calculation where the TF and the IDF scores are first calculated which is then multiplied to get the final score. TF (which stands for term frequency) of a term *t* is calculated by dividing the count vector of *t* in the document (d_x) by the number of documents in the entire corpus (d) with term *t*. IDF (which stands for inverse document frequency) of a term *t* is computed as the logarithm value of the total number of the documents in the corpus divided by the total number of documents in the entire corpus (d) with term *t*. [12]

```
TF-IDF Vectorizer Array:
[[0.40572238 0.         0.         0.         0.
  0.91399636]
 [0.35748457 0.66038049 0.         0.         0.66038049 0.
  0.        ]
 [0.31223453 0.57679018 0.4869659  0.         0.         0.57679018
  0.        ]
 [0.35748457 0.         0.         0.66038049 0.66038049 0.
  0.        ]
 [0.31223453 0.         0.4869659  0.57679018 0.         0.57679018
  0.        ]
 [0.53976033 0.         0.84181874 0.         0.         0.
  0.        ]]
TF-IDF Vectorizer size :  6
TF-IDF Vectorizer feature names:
['comment', 'field', 'line', 'method', 'multilin', 'singl', 'test']
```

Fig. 4. Count vector representation in NLTK

$$TF(t, d, d_x) = count(t, d_x)/count(t, d) \tag{1}$$

$$IDF(t, d) = log_e(count(d)/count(t, d)) \tag{2}$$

$$TF - IDF(t, d) = TF(t, d, d_x) * IDF(t, d) \tag{3}$$

The term *t* in the TF-IDF score can be scoped into three levels; word level, n-gram level and character level. When the TF-IDF score is calculated considering the word level, it examines each word in all the documents. The n-gram level considers *n* number of words determined by the user. Finally, at the character level, every character is recognised for the calculation. However, it seems that setting the scope depending on the size of the dataset yields better results. Character level should be considered with a small number of documents, while n-gram level should be considered with a more significant corpus.

TABLE II
TF-IDF SCOPING AND CLUSTER RESULTS

TF-IDF scope level	**Corpus size**	**Number of clusters**	**Average documents per cluster**
Word	14,243	1,354	10.5
N-gram (2)	14,243	654	21.71
N-gram (3)	14,243	432	32.81
N-gram (4)	14,243	371	38.18
N-gram (5)	14,243	332	42.64

V. Discussion

The DBSCAN algorithm was used to cluster a sequence of comment documents. During the initial stage, the number of clusters is unknown. DBSCAN can cluster the documents by analysing the cluster density specified by the neighbourhood size. It creates an arbitrary number of clusters depending on the available data. Since the initial number of clusters is not available or restricted to a predetermined value, the algorithm can analyse the high-density areas to produce uneven and non-flat geometry cluster sizes. DBSCAN is also resistant to noise and robust to outliers. [12]

A. Accumulated noise

One crucial matter I noticed in the project was that the algorithm labels more data as noise if the dataset is small. This behaviour depends on the *min_samples* variable which determines the minimum documents required to form a cluster. The higher *min_samples* value, higher the number of documents classified as noise. If the *min_samples* value is 1, lots of clusters are generated with one document, which is expected since there aren't enough data in a small dataset. This phenomenon declines will the corpus size. Nearly 84% of the documents in a corpus with around 3,000 documents were classified as noise. However, only 72% were classified as noise in a corpus of size 14,000.

TABLE III
NOISE VS CORPUS SIZE (*min_sample*=2)

Corpus size	Noise %
1,231	95%
2,156	89%
3,564	84%
4,675	81%
7,435	79%
13,923	72%
20,328	55%

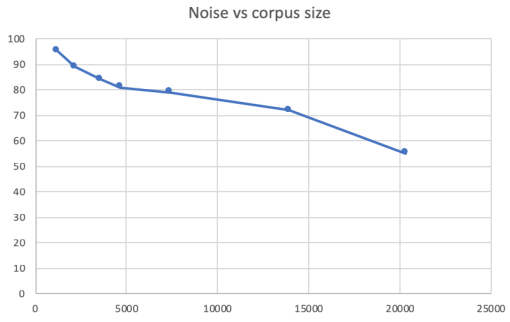

Fig. 5. Noise vs corpus size

Fig. 5 shows that the noise value decreases when a more significant corpus is used. This is expected because it should have access to a bigger dataset for the algorithm to form meaningful clusters. Most notably, the noise percentage started decreasing in an exponential rate when the corpus size is increased. In the range of size 1,000 to 4,000, it had a slow decline. However, it started plummeting when the corpus size was gradually increased up to 20,000. This must be because the algorithm has more data to form more clusters which otherwise would have labelled as noise. Since *min_samples* was set to 2, there wasn't enough data to create a valid cluster when the corpus size was below 5,000.

B. Cluster accuracy and cohesion

The other parameter *eps* value determines the similarity between two different samples. The accuracy of the classification mainly depends on this value. Less value indicates higher cohesion between the clusters and a higher value indicates a lower coherence. It is advised to determine this parameter depending on the dataset. By testing the algorithm with the obtained comment data (corpus of size nearly 14,000), it seemed that 0.7 is a better value for balancing the cluster cohesion and the validity.

TABLE IV
eps VALUE VS CLUSTER FORMATION

eps value	Corpus size	Number of clusters	Average documents per cluster
0.1	14,252	26	4.4
0.2	14,252	31	4.2
0.3	14,252	41	4.0
0.4	14,252	83	4.16
0.5	14,252	154	4.2
0.6	14,252	258	4.3
0.7	14,252	398	4.42
0.8	14,252	561	4.78
0.9	14,252	728	5.60

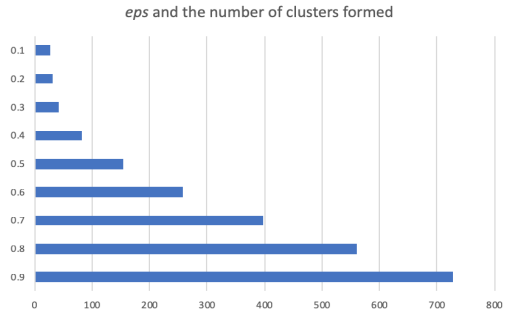

Fig. 6. Noise vs corpus size

The attached Appendix B lists examples of the clusters formed with the *eps* value of 0.7 and *min_sample* of 3.

VI. Evaluation

A. Threats to validity

As described in Section V, the clustering performance and accuracy is improved with a bigger corpus size. The more data

the algorithm has access to, better the results are. However, text classification seemed a very resource intensive task which requires decent computing power. The algorithm was tested with a comment set of around 20,000 comments obtained from the Android Open Source Project, Firebase Admin API and Apache Hadoop Project. A typical computer with Dual-Core i5 processor and 8GB of RAM takes around 1 hour to process the given 20,000 comments. Using a variety of projects with a lot more comment data would produce a better-unbiased results set.

B. Limitations

One of the main limitations of the project is that it can only capture comments in Java source files. Additionally, the Java source file name should end with the *.java* file extension. The program will omit all other files in the resources directory. This limitation dramatically restricts the application only to Java development teams. Even though not tested, some other Java-like languages such as Scala and Kotlin which also use the same commenting syntax should work with the program. However, this would still need a minor modification to allow the Scala and Kotlin files to be scanned by whitelisting the respective file extensions.

Another limitation of this project is that the scanned corpus is limited to the amount of installed RAM in the running computer. Currently, all scanned data is processed while accessing it from the main memory. Even though this decision significantly reduces the execution time, it also limits the number of comments that the computer can handle. Due to this constraint, the program requires a significant amount of RAM to process extensive datasets.

The program attempts to remove one to two-word comments which do not provide a context value. Also, licence text which usually appears at the beginning of the source file is also omitted. However, some developers tend to comment unused code which can affect the clustering process if done excessively. If the unused piece of code is commented using single-line comments, those will be interpreted as several single-line comments. For example, if Eclipse and JetBrains IDE is used to comment out unused code, the IDE comments each line as a single-line comment. This can affect the classification if this unused commented out code is lengthy since it will produce one comment per each line. Also, some developers may use single-line commenting syntax to write multi-line comments. In this scenario, unfortunately, the program will pick up each line as a separate comment thereby affecting the clustering process.

The program can analyse and cluster the comments written in English only. Even though technically it can detect comments written in other languages, the accuracy of the output is undefined and untested. Several places in the program are hard-coded to support English language only. In case if non-English comments are present in the scanned documents, those comments will not be omitted but will be processed as if it were in English.

C. Future Work

1) Improving the model for supervised learning: This project attempts to cluster a given sequence of comment data. As described in SECTION I, this technique is known as unsupervised learning. As opposed to the unsupervised approach, the supervised learning approach uses a pre-clustered set of documents to model its dataset. The clusters formed by this project can be used as a model for future supervised learning techniques to predict and identify more comment patterns that exist in the current software projects.

2) Performance improvements: The program requires a computer with reasonable processing power to process an adequate amount of data. The main focus of this project was to develop a program to collect comments in a source file and to cluster them. The proposed software can be optimised further to perform document clustering with limited resources. The program utilises around 5-6GB of RAM to handle a document set of approximately 20,000 documents. Currently, all comment data and generated Count and TF-IDF vectors are stored in the primary memory (RAM). This data can be cached elsewhere to release RAM memory for other tasks. In addition, using more memory efficient data structures (like Pandas Series) may also increase the memory efficiency.

3) Extending to other languages: The proposed source code scanner only works with Java source files with the file extension *.java*. This can be improved furthermore to capture the comments in other languages as well. Adopting the algorithm for Java-like languages which uses the same lexical structure for the comments should be a simple modification to enable the algorithm to process different file extensions. *JavaFileInspector* class (class responsible for collecting comments from a Java source file) can be modified for other languages which use a completely different syntax. It is implemented as a Python class, and by utilising OOP concepts, we can extend the same class to provide support for other languages.

REFERENCES

[1] Z. Faguo, Z. Fan, Y. Bingru, and Y. Xingang, "Research on short text classification algorithm based on statistics and rules," in *2010 Third International Symposium on Electronic Commerce and Security*, July 2010, pp. 3–7.

[2] H. Isahara, "Resource-based natural language processing," in *2007 International Conference on Natural Language Processing and Knowledge Engineering*, Aug 2007, pp. 11–12.

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*, ser. Springer Texts in Statistics. Springer, 2017, no. DOI 10.1007/978-1-4614-7138-7.

[4] G. J. Badros, "Javaml: a markup language for java source code," *Computer Networks*, vol. 33, no. 1, pp. 159 – 177, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128600000372

[5] IBM, "Ibm jikes java compiler framework." [Online]. Available: http://jikes.sourceforge.net

[6] R. Koschke, R. Falke, and P. Frenzel, "Clone detection using abstract syntax suffix trees," in *2006 13th Working Conference on Reverse Engineering*, Oct 2006, pp. 253–262.

[7] C. Thunes, "Pure python java parser." [Online]. Available: https://github.com/c2nes/javalang

[8] Q. Li and X. Huang, "Research on text clustering algorithms," in *2010 2nd International Workshop on Database Technology and Applications*, Nov 2010, pp. 1–3.

[9] X.-W. Liu, P.-L. He, and H.-Y. Wang, "The research of text clustering algorithms based on frequent term sets," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 4, Aug 2005, pp. 2352–2356 Vol. 4.

[10] Z. Yuhang, W. Yue, and Y. Wei, "Research on data cleaning in text clustering," in *2010 International Forum on Information Technology and Applications*, vol. 1, July 2010, pp. 305–307.

[11] P. Han, S. Shen, D. Wang, and Y. Liu, "The influence of word normalization in english document clustering," in *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, vol. 2, May 2012, pp. 116–120.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix A - Readme file

Clustering Source code comments

System Requirements

- Python 3.6 (pythonw (Python GUI) is required to view the plot results)
- pip3 (for installing python dependent modules)

Python Dependencies

- nltk
- pandas
- sklearn
- colorma
- javalang
- python-dateutil

Preparing Environment

- Unzip or clone (`git clone https://github.com/wsimf/classify_source_code_comments.git`) the project to a convenient location
- Open the command prompt / terminal and navigate to the project folder
- Make sure Python3 is installed (`python -V`)
- Install pythonw (if you are using anaconda for python environment management, you can use `conda install python.app` on a macOS environment)
- Install dependencies (`pip install -r requirements.txt`)

Running the application

- Copy Java source files into `/comment_resources` directory
- Run `pythonw comment_analyser.py`
- All the found comments will be written into `comments.txt` file
- The clustered result will be written into `comments_cluster_dbscan.txt` file
- The count and TF-IDF vectorizer results will open in a seperate window

Python GUI (plot results) were tested on a macOS environment. The `pythonw` should be installed depending on the operating system used. Not all operating systems support python GUI. The program would fail with an error if it was unable to draw the GUI component. However, the clustering and the generation of `comments.txt` and `comments_cluster_dbscan.txt` will successfully complete irrespective of this issue.

APPENDIX B - CLUSTER RESULT 01

Some parts of the cluster results file is attached below. For the full result list, please refer to the `comments_cluster_dbscan.txt` file committed to the code repository.

```
** [CLUSTER] : 0
      [COMMENT] : Verify the LOHS onStarted callback is triggered when WifiManager receives a HOTSPOT_STARTED message from
WifiServiceImpl. /
      [COMMENT] : Verify onStopped is called if WifiServiceImpl sends a HOTSPOT_STOPPED message. /
      [COMMENT] : Verify the LOHS onRegistered observer callback is triggered when WifiManager receives a HOTSPOT_OBSERVER_REGISTERED
message from WifiServiceImpl. /
      [COMMENT] : Verify the LOHS onStarted observer callback is triggered when WifiManager receives a HOTSPOT_STARTED message from
WifiServiceImpl. /
      [COMMENT] : Verify the LOHS onStarted observer callback is triggered not when WifiManager receives a HOTSPOT_STARTED message from
WifiServiceImpl with a null config. /
      [COMMENT] : Verify the LOHS onStopped observer callback is triggered when WifiManager receives a HOTSPOT_STOPPED message from
WifiServiceImpl. /
      [COMMENT] : Verify WifiServiceImpl is not called if there is not a registered LOHS observer callback. /
      [COMMENT] : Verify WifiServiceImpl is called when there is a registered LOHS observer callback. /

** [CLUSTER] : 1
      [COMMENT] : Verify that the call to cancel a LOHS request does call stopLOHS. /
      [COMMENT] : Verify that the callback is not triggered if the LOHS request was already cancelled. /
      [COMMENT] : Verify that calling cancel LOHS request does not crash if an error callback was already handled. /

** [CLUSTER] : 2
      [COMMENT] : Validate the successful connect flow: (1) connect + success (2) publish, (3) disconnect (4) try publishing on old
session (5) connect again /
      [COMMENT] : Validate the publish flow: (0) connect + success, (1) publish, (2) success creates session, (3) pass through everything,
(4) update publish through session, (5) terminate locally, (6) try another command - ignored. /
      [COMMENT] : Validate the subscribe flow: (0) connect + success, (1) subscribe, (2) success creates session, (3) pass through
everything, (4) update subscribe through session, (5) terminate locally, (6) try another command - ignored. /
      [COMMENT] : Validate ranging + success flow: (1) connect, (2) create a (publish) session, (3) start ranging, (4) ranging success
callback, (5) ranging aborted callback ignored (since listener removed). /

** [CLUSTER] : 3
      [COMMENT] : Validate that an empty agent network specifier doesn't match any base network specifier. /
      [COMMENT] : Validate that an agent network specifier constructed with a single entry matches that entry, and only that entry. /
      [COMMENT] : Validate that an agent network specifier constructed with multiple entries matches all those entries - but none other. /
      [COMMENT] : Validate that agent network specifier does not match against a sub-set. /

** [CLUSTER] : 5
      [COMMENT] : Helper function for creating a {@link Policy} for testing.  @return {@link Policy} /
      [COMMENT] : Helper function for creating a {@link PasspointConfiguration} for testing.  @return {@link PasspointConfiguration} /
      [COMMENT] : Helper function for generating certificate credential for testing.  @return {@link Credential} /
      [COMMENT] : Helper function for generating SIM credential for testing.  @return {@link Credential} /
      [COMMENT] : Helper function for generating user credential for testing.  @return {@link Credential} /
      [COMMENT] : Helper function for creating a {@link UpdateParameter} for testing.  @return {@link UpdateParameter} /

** [CLUSTER] : 6
      [COMMENT] : Verify that copy constructor works when pass in a null source.  @throws Exception /
      [COMMENT] : Verify that copy constructor works when pass in a valid source.  @throws Exception /
      [COMMENT] : Verify that copy constructor works when pass in a source with user credential.  @throws Exception /
      [COMMENT] : Verify that copy constructor works when pass in a source with certificate credential.  @throws Exception /
      [COMMENT] : Verify that copy constructor works when pass in a source with SIM credential.  @throws Exception /
      [COMMENT] : Verify that policy created using copy constructor with null source should be the same as the policy created using
default constructor.  @throws Exception /
      [COMMENT] : Verify that policy created using copy constructor with a valid source should be the same as the source.  @throws
Exception /
      [COMMENT] : Verify that a policy created using {@link #createPolicy} is valid, since all fields are filled in with valid values.
@throws Exception /
      [COMMENT] : Verify that UpdateParameter created using copy constructor with null source should be the same as the UpdateParameter
created using default constructor.  @throws Exception /
      [COMMENT] : Verify that UpdateParameter created using copy constructor with a valid source should be the same as the source.
@throws Exception /
      [COMMENT] : Verify that an UpdateParameter created using {@link #createUpdateParameter} is valid, since all fields are filled in
with valid values.  @throws Exception /

** [CLUSTER] : 7
      [COMMENT] : Verify a valid installation file is parsed successfully with the matching contents.  @throws Exception /
      [COMMENT] : Verify that parsing an installation file with invalid MIME type will fail.  @throws Exception /
      [COMMENT] : Verify that parsing an un-encoded installation file will fail.  @throws Exception /
      [COMMENT] : Verify that parsing an installation file that contains a non-base64 part will fail.  @throws Exception /
      [COMMENT] : Verify that parsing an installation file that contains a missing boundary string will fail.  @throws Exception /
      [COMMENT] : Verify that parsing an installation file that contains a MIME part with an invalid content type will fail.  @throws
Exception /
      [COMMENT] : Verify that parsing an installation file that doesn't contain a Passpoint profile will fail.  @throws Exception /

** [CLUSTER] : 8
      [COMMENT] : Helper function for creating a map of home network IDs for testing.  @return Map of home network IDs /
      [COMMENT] : Helper function for creating a HomeSp for testing.  @param homeNetworkIds The map of home network IDs associated with


** [CLUSTER] : 4
      [COMMENT] :  Setting the client certificate to null should clear the existing chain.
      [COMMENT] : Verify parcel read/write for an OSU provider containing no information.  @throws Exception /
      [COMMENT] : Verify parcel read/write for an OSU provider containing full information.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that contained the full configuration.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain HomeSP.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain Credential.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain Policy.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain subscription update.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain trust root certificate list.  @throws Exception /
      [COMMENT] : Verify that a configuration without Credential is invalid.  @throws Exception /
      [COMMENT] : Verify that a configuration without HomeSP is invalid.  @throws Exception /
      [COMMENT] : Verify that a configuration with a trust root certificate URL exceeding the max size is invalid.  @throws Exception /
      [COMMENT] : Verify that a configuration with an invalid trust root certificate fingerprint is invalid.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a HomeSp containing Home Network IDs.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a HomeSp without Home Network IDs.  @throws Exception /
```

[COMMENT] : Verify that a HomeSp is valid when both FQDN and Friendly Name are provided. @throws Exception /
[COMMENT] : Verify that a HomeSp is not valid when FQDN is not provided @throws Exception /
[COMMENT] : Verify that a HomeSp is not valid when Friendly Name is not provided @throws Exception /
[COMMENT] : Verify that a HomeSp is valid when the optional Home Network IDs are not provided. @throws Exception /
[COMMENT] : Verify that a HomeSp is invalid when the optional Home Network IDs contained an invalid SSID (exceeding maximum number of bytes). @throws Exception /
[COMMENT] : Verify that an user credential without CA Certificate is invalid. @throws Exception /
[COMMENT] : Verify that an user credential with EAP type other than EAP-TTLS is invalid. @throws Exception /
[COMMENT] : Verify that an user credential without realm is invalid. @throws Exception /
[COMMENT] : Verify that an user credential without username is invalid. @throws Exception /
[COMMENT] : Verify that an user credential without password is invalid. @throws Exception /
[COMMENT] : Verify that an certificate credential without CA Certificate is invalid. @throws Exception /
[COMMENT] : Verify that a certificate credential without client certificate chain is invalid. @throws Exception /
[COMMENT] : Verify that a certificate credential without client private key is invalid. @throws Exception /
[COMMENT] : Verify that a certificate credential with mismatch client certificate fingerprint is invalid. @throws Exception /
[COMMENT] : Verify that a SIM credential without IMSI is invalid. @throws Exception /
[COMMENT] : Verify that a SIM credential with an invalid IMSI is invalid. @throws Exception /
[COMMENT] : Verify that a SIM credential with invalid EAP type is invalid. @throws Exception /
[COMMENT] : Verify that a credential contained both an user and a SIM credential is invalid. @throws Exception /
[COMMENT] : Verify parcel read/write for a Policy with all fields set. @throws Exception /
[COMMENT] : Verify parcel read/write for a Policy without protocol port map. @throws Exception /
[COMMENT] : Verify parcel read/write for a Policy without preferred roaming partner list. @throws Exception /
[COMMENT] : Verify parcel read/write for a Policy without policy update parameters. @throws Exception /
[COMMENT] : Verify that a default policy (with no informatio) is invalid. @throws Exception /
[COMMENT] : Verify that a policy without policy update parameters is invalid. @throws Exception /
[COMMENT] : Verify that a policy with invalid policy update parameters is invalid. @throws Exception /
[COMMENT] : Verify that a policy with a preferred roaming partner with FQDN not specified is invalid. @throws Exception /
[COMMENT] : Verify that a policy with a preferred roaming partner with countries not specified is invalid. @throws Exception /
[COMMENT] : Verify that a policy with number of excluded SSIDs exceeded the max is invalid. @throws Exception /
[COMMENT] : Verify that a policy with an invalid SSID in the excluded SSID list is invalid. @throws Exception /
[COMMENT] : Verify parcel read/write for a UpdateParameter with all fields set. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an unknown update method is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an unknown restriction is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an username exceeding maximum size is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an empty username is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with a password exceeding maximum size is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an empty password is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with a Base64 encoded password that contained invalid padding is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter without trust root certificate URL is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with invalid trust root certificate URL is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter without trust root certificate SHA-256 fingerprint is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an incorrect size trust root certificate SHA-256 fingerprint is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter without server URI is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an invalid server URI is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with update interval set to "never" will not perform validation on other parameters, since update is not applicable in this case. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with unset update interval is invalid. @throws Exception /
[COMMENT] : Set realm for Passpoint credential; realm identifies a set of networks where your Passpoint credential can be used @param realm the realm /
[COMMENT] : Get realm for Passpoint credential; see {@link #setRealm(String)} for more information @return the realm /
[COMMENT] : Set the credential information. @param credential The credential information to set to /
[COMMENT] : Credential is needed for storing the certificates and private client key.
[COMMENT] : Set the realm associated with this credential. @param realm The realm to set to /
[COMMENT] : Get the realm associated with this credential. @return the realm associated with this credential /
[COMMENT] : Set the username associated with this user credential. @param username The username to set to /
[COMMENT] : Get the username associated with this user credential. @return the username associated with this user credential /
[COMMENT] : Set the user credential information. @param userCredential The user credential to set to /
[COMMENT] : Set the certificate type associated with this certificate credential. @param certType The certificate type to set to /

** [CLUSTER] : 11
[COMMENT] : Broadcast intent action indicating whether Wi-Fi scanning is allowed currently @hide /
[COMMENT] : Broadcast intent action indicating that Wi-Fi AP has been enabled, disabled, enabling, disabling, or failed. @hide /
[COMMENT] : Broadcast intent action indicating that the link configuration changed on wifi. @hide /
[COMMENT] : Broadcast intent action indicating that this device details have changed. /
[COMMENT] : Broadcast intent action indicating that remembered persistent groups have changed. @hide /

** [CLUSTER] : 12
[COMMENT] : Broadcast intent action indicating that a Passpoint provider icon has been received. Included extras: {@link #EXTRA_BSSID_LONG} {@link #EXTRA_FILENAME} {@link #EXTRA_ICON} Receiver Required Permission: android.Manifest.permission.ACCESS_WIFI_STATE <p>Note: The broadcast is only delivered to registered receivers - no manifest registered components will be launched. @hide /
[COMMENT] : Broadcast intent action indicating a Passpoint OSU Providers List element has been received. Included extras: {@link #EXTRA_BSSID_LONG} {@link #EXTRA_ANQP_ELEMENT_DATA} Receiver Required Permission: android.Manifest.permission.ACCESS_WIFI_STATE <p>Note: The broadcast is only delivered to registered receivers - no manifest registered components will be launched. @hide /
[COMMENT] : Broadcast intent action indicating that a Passpoint Deauth Imminent frame has been received. Included extras: {@link #EXTRA_BSSID_LONG} {@link #EXTRA_ESS} {@link #EXTRA_DELAY} {@link #EXTRA_URL} Receiver Required Permission: android.Manifest.permission.ACCESS_WIFI_STATE <p>Note: The broadcast is only delivered to registered receivers - no manifest registered components will be launched. @hide /
[COMMENT] : Broadcast intent action indicating a Passpoint subscription remediation frame has been received. Included extras: {@link #EXTRA_BSSID_LONG} {@link #EXTRA_SUBSCRIPTION_REMEDIATION_METHOD} {@link #EXTRA_URL} Receiver Required Permission: android.Manifest.permission.ACCESS_WIFI_STATE <p>Note: The broadcast is only delivered to registered receivers - no manifest registered components will be launched. @hide /

** [CLUSTER] : 13
[COMMENT] : The lookup key for an int that indicates whether Wi-Fi is enabled, disabled, enabling, disabling, or unknown. Retrieve it with {@link android.content.Intent#getIntExtra(String,int)}. @see #WIFI_STATE_DISABLED @see #WIFI_STATE_DISABLING @see #WIFI_STATE_ENABLED @see #WIFI_STATE_ENABLING @see #WIFI_STATE_UNKNOWN /
[COMMENT] : The lookup key for an int that indicates whether Wi-Fi AP is enabled, disabled, enabling, disabling, or failed. Retrieve it with {@link android.content.Intent#getIntExtra(String,int)}. @see #WIFI_AP_STATE_DISABLED @see #WIFI_AP_STATE_DISABLING @see #WIFI_AP_STATE_ENABLED @see #WIFI_AP_STATE_ENABLING @see #WIFI_AP_STATE_FAILED @hide /
[COMMENT] : Gets the Wi-Fi enabled state. @return One of {@link #WIFI_STATE_DISABLED}, {@link #WIFI_STATE_DISABLING}, {@link #WIFI_STATE_ENABLED}, {@link #WIFI_STATE_ENABLING}, {@link #WIFI_STATE_UNKNOWN} @see #isWifiEnabled() /
[COMMENT] : Gets the Wi-Fi enabled state. @return One of {@link #WIFI_AP_STATE_DISABLED}, {@link #WIFI_AP_STATE_DISABLING}, {@link #WIFI_AP_STATE_ENABLED}, {@link #WIFI_AP_STATE_ENABLING}, {@link #WIFI_AP_STATE_FAILED} @see #isWifiApEnabled() @hide /
[COMMENT] : The lookup key for an int that indicates whether Wi-Fi p2p is enabled or disabled. Retrieve it with {@link android.content.Intent#getIntExtra(String,int)}. @see #WIFI_P2P_STATE_DISABLED @see #WIFI_P2P_STATE_ENABLED /
[COMMENT] : The lookup key for an int that indicates whether p2p discovery has started or stopped. Retrieve it with {@link android.content.Intent#getIntExtra(String,int)}. @see #WIFI_P2P_DISCOVERY_STARTED @see #WIFI_P2P_DISCOVERY_STOPPED /

** [CLUSTER] : 14
[COMMENT] : Wi-Fi AP is currently being disabled. The state will change to {@link #WIFI_AP_STATE_DISABLED} if it finishes successfully. @see #WIFI_AP_STATE_CHANGED_ACTION @see #getWifiApState() @hide /
[COMMENT] : Wi-Fi AP is currently being enabled. The state will change to {@link #WIFI_AP_STATE_ENABLED} if it finishes successfully. @see #WIFI_AP_STATE_CHANGED_ACTION @see #getWifiApState() @hide /
[COMMENT] : Wi-Fi AP is in a failed state. This state will occur when an error occurs during enabling or disabling @see #WIFI_AP_STATE_CHANGED_ACTION @see #getWifiApState() @hide /
[COMMENT] : Return whether Wi-Fi AP is enabled or disabled. @return {@code true} if Wi-Fi AP is enabled @see #getWifiApState() @hide /

** [CLUSTER] : 15
	[COMMENT] : The lookup key for a {@link android.net.NetworkInfo} object associated with the Wi-Fi network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a String giving the BSSID of the access point to which we are connected. Only present when the new state is CONNECTED. Retrieve with {@link android.content.Intent#getStringExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.wifi.WifiInfo} object giving the information about the access point to which we are connected. Only present when the new state is CONNECTED. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link SupplicantState} describing the new state Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.LinkProperties} object associated with the Wi-Fi network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. @hide /
	[COMMENT] : The lookup key for a {@link android.net.NetworkCapabilities} object associated with the Wi-Fi network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. @hide /
	[COMMENT] : The lookup key for a {@link android.net.wifi.p2p.WifiP2pInfo} object Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.NetworkInfo} object associated with the p2p network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.wifi.p2p.WifiP2pGroup} object associated with the p2p network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.wifi.p2p.WifiP2pDevice} object Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /

** [CLUSTER] : 16
	[COMMENT] : A batch of access point scans has been completed and the results areavailable. Call {@link #getBatchedScanResults()} to obtain the results. @deprecated This API is nolonger supported. Use {@link android.net.wifi.WifiScanner} API @hide /
	[COMMENT] : Check if the Batched Scan feature is supported. @return false if not supported. @deprecated This API is nolonger

** [CLUSTER] : 20
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed due to an internal error. @hide /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation is already in progress @hide /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed due to invalid inputs @hide /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed due to user permissions. @hide /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed due to an internal error. /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed because p2p is unsupported on the device. /

** [CLUSTER] : 21
	[COMMENT] : WPS operation failed @param reason The reason for failure could be one of {@link #WPS_TKIP_ONLY_PROHIBITED}, {@link #WPS_OVERLAP_ERROR}, {@link #WPS_WEP_PROHIBITED}, {@link #WPS_TIMED_OUT} or {@link #WPS_AUTH_FAILURE} and some generic errors. /
	[COMMENT] : The operation failed @param reason The reason for failure could be one of {@link #ERROR}, {@link #IN_PROGRESS} or {@link #BUSY} /
	[COMMENT] : The operation failed @param reason The reason for failure could be one of {@link #P2P_UNSUPPORTED}, {@link #ERROR} or {@link #BUSY} /

** [CLUSTER] : 22
	[COMMENT] : Connect to a network with the given configuration. The network also gets added to the list of configured networks for the foreground user. For a new network, this function is used instead of a sequence of addNetwork(), enableNetwork(), saveConfiguration() and reconnect() @param config the set of variables that describe the configuration, contained in a {@link WifiConfiguration} object. @param listener for callbacks on success or failure. Can be null. @throws IllegalStateException if the WifiManager instance needs to be initialized again @hide /
	[COMMENT] : Connect to a network with the given networkId. This function is used instead of a enableNetwork(), saveConfiguration() and reconnect() @param networkId the ID of the network as returned by {@link #addNetwork} or {@link getConfiguredNetworks}. @param listener for callbacks on success or failure. Can be null. @throws IllegalStateException if the WifiManager instance needs to be initialized again @hide /
	[COMMENT] : Save the given network to the list of configured networks for the foreground user. If the network already exists, the configuration is updated. Any new network is enabled by default. For a new network, this function is used instead of a sequence of addNetwork(), enableNetwork() and saveConfiguration(). For an existing network, it accomplishes the task of updateNetwork() and saveConfiguration() @param config the set of variables that describe the configuration, contained in a {@link WifiConfiguration} object. @param listener for callbacks on success or failure. Can be null. @throws IllegalStateException if the WifiManager instance needs to be initialized again @hide /
	[COMMENT] : Delete the network from the list of configured networks for the foreground user. This function is used instead of a sequence of removeNetwork() and saveConfiguration(). @param config the set of variables that describe the configuration, contained in a {@link WifiConfiguration} object. @param listener for callbacks on success or failure. Can be null. @throws IllegalStateException if the WifiManager instance needs to be initialized again @hide /

** [CLUSTER] : 23
	[COMMENT] : Get a reference to WifiService handler. This is used by a client to establish an AsyncChannel communication with WifiService @return Messenger pointing to the WifiService handler @hide /
	[COMMENT] : Get a reference to WifiP2pService handler. This is used to establish an AsyncChannel communication with WifiService @param binder A binder for the service to associate with this client. @return Messenger pointing to the WifiP2pService handler @hide /
	[COMMENT] : Get a reference to P2pStateMachine handler. This is used to establish a priveleged AsyncChannel communication with WifiP2pService. @return Messenger pointing to the WifiP2pService handler @hide /

** [CLUSTER] : 24
	[COMMENT] : Locks the Wi-Fi radio on until {@link #release} is called. If this WifiLock is reference-counted, each call to {@code acquire} will increment the reference count, and the radio will remain locked as long as the reference count is above zero. If this WifiLock is not reference-counted, the first call to {@code acquire} will lock the radio, but subsequent calls will be ignored. Only one call to {@link #release} will be required, regardless of the number of times that {@code acquire} is called. /
	[COMMENT] : Unlocks the Wi-Fi radio, allowing it to turn off when the device is idle. If this WifiLock is reference-counted, each call to {@code release} will decrement the reference count, and the radio will be unlocked only when the reference count reaches zero. If the reference count goes below zero (that is, if {@code release} is called a greater number of times than {@link #acquire}), an exception is thrown. If this WifiLock is not reference-counted, the first call to {@code release} (after the radio was locked using {@link #acquire}) will unlock the radio, and subsequent calls will be ignored. /
	[COMMENT] : Locks Wifi Multicast on until {@link #release} is called. If this MulticastLock is reference-counted each call to {@code acquire} will increment the reference count, and the wifi interface will receive multicast packets as long as the reference count is above zero. If this MulticastLock is not reference-counted, the first call to {@code acquire} will turn on the multicast packets, but subsequent calls will be ignored. Only one call to {@link #release} will be required, regardless of the number of times that {@code acquire} is called. Note that other applications may also lock Wifi Multicast on. Only they can relinquish their lock. Also note that applications cannot leave Multicast locked on. When an app exits or crashes, any Multicast locks will be released. /
	[COMMENT] : Unlocks Wifi Multicast, restoring the filter of packets not addressed specifically to this device and saving power. If this MulticastLock is reference-counted, each call to {@code release} will decrement the reference count, and the multicast packets will only stop being received when the reference count reaches zero. If the reference count goes below zero (that is, if {@code release} is called a greater number of times than {@link #acquire}), an exception is thrown. If this MulticastLock is not reference-counted, the first call to {@code release} (after the radio was multicast locked using {@link #acquire}) will unlock the multicast, and subsequent calls will be ignored. Note that if any other Wifi Multicast Locks are still outstanding this {@code release} call will not have an immediate effect. Only when all applications have released all their Multicast Locks will the Multicast filter be turned back on. Also note that when an app exits or crashes all of its Multicast Locks will be automatically released. /

** [CLUSTER] : 25
	[COMMENT] : Creates a new WifiLock. @param lockType the type of lock to create. See {@link #WIFI_MODE_FULL}, {@link #WIFI_MODE_FULL_HIGH_PERF} and {@link #WIFI_MODE_SCAN_ONLY} for descriptions of the types of Wi-Fi locks. @param tag a tag for the WifiLock to identify it in debugging messages. This string is never shown to the user under normal conditions, but should be descriptive enough to identify your application and the specific WifiLock within it, if it holds multiple WifiLocks. @return a new, unacquired

** [CLUSTER] : 29
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_20_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_40_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_80_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_160_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_5_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_10_SUPPORT} API. /

** [CLUSTER] : 30
	[COMMENT] : Not used if the AP bandwidth is 20 MHz If the AP use 40, 80 or 160 MHz, this is the center frequency if the AP use 80 + 80 MHz, this is the center frequency of the first segment same as ScanResult.centerFreq0 Default value: 0 /
	[COMMENT] : Only used if the AP bandwidth is 80 + 80 MHz if the AP use 80 + 80 MHz, this is the center frequency of the second segment same as ScanResult.centerFreq1 Default value: 0 /
	[COMMENT] : Center frequency of the second segment when channel width is 80 + 80 MHz. @see ScanResult#centerFreq1 /
	[COMMENT] : AP Channel bandwidth is 160 MHZ, but 80MHZ + 80MHZ /
	[COMMENT] : AP Channel bandwidth; one of {@link #CHANNEL_WIDTH_20MHZ}, {@link #CHANNEL_WIDTH_40MHZ}, {@link #CHANNEL_WIDTH_80MHZ}, {@link #CHANNEL_WIDTH_160MHZ} or {@link #CHANNEL_WIDTH_80MHZ_PLUS_MHZ}. /
	[COMMENT] : Not used if the AP bandwidth is 20 MHz If the AP use 40, 80 or 160 MHz, this is the center frequency (in MHz) if the AP use 80 + 80 MHz, this is the center frequency of the first segment (in MHz) /
	[COMMENT] : Only used if the AP bandwidth is 80 + 80 MHz if the AP use 80 + 80 MHz, this is the center frequency of the second segment (in MHz) /

** [CLUSTER] : 31
	[COMMENT] : RSSI spread (i.e. max - min) @deprecated Use {@link android.net.wifi.RttManager.RttResult#rssiSpread} API. /
	[COMMENT] : spread (i.e. max - min) round trip time @deprecated Use {@link android.net.wifi.RttManager.RttResult#rttSpread} API. /
	[COMMENT] : spread (i.e. max - min) distance @deprecated Use {@link android.net.wifi.RttManager.RttResult#distanceSpread} API. /

** [CLUSTER] : 42
	[COMMENT] : The current status of this network configuration entry. @see Status /
	[COMMENT] : Quality network selection status String (for debug purpose). Use Quality network selection status value as index to extec the corresponding debug string /
	[COMMENT] : get current Quality network selection status @return returns current Quality network selection status in String (for debug purpose) /
	[COMMENT] : @param reason specific error reason @return corresponding network disable reason String (for debug purpose) /
	[COMMENT] : get current network disable reason @return current network disable reason in String (for debug purpose) /
	[COMMENT] : get current network network selection status @return return current network network selection status /
	[COMMENT] : set current networ work selection status @param status network selection status to set /
	[COMMENT] : BSSID for connection to this network (through network selection procedure) /
	[COMMENT] : get current network Selection BSSID @return current network Selection BSSID /
	[COMMENT] : set network Selection BSSID @param bssid The target BSSID for assocaition /

** [CLUSTER] : 37
	[COMMENT] : The network's SSID. Can either be an ASCII string, which must be enclosed in double quotation marks (e.g., {@code "MyNetwork"}, or a string of hex digits,which are not enclosed in quotes (e.g., {@code 01a243f405}). /
	[COMMENT] : The network's SSID. Can either be an ASCII string, which must be enclosed in double quotation marks (e.g., {@code "MyNetwork"}), or a string of hex digits, which are not enclosed in quotes (e.g., {@code 01a243f405}). /
	[COMMENT] : Pre-shared key for use with WPA-PSK. Either an ASCII string enclosed in double quotation marks (e.g., {@code "abcdefghij"} for PSK passphrase or a string of 64 hex digits for raw PSK. When the value of this key is read, the actual key is not returned, just a "*" if the key has a value, or the null string otherwise. /
	[COMMENT] : Up to four WEP keys. Either an ASCII string enclosed in double quotation marks (e.g., {@code "abcdef"}) or a string of hex digits (e.g., {@code 0102030405}). When the value of one of these keys is read, the actual key is not returned, just a "*" if the key has a value, or the null string otherwise. /

** [CLUSTER] : 32
	[COMMENT] : WME Best Effort Access Category (receive mpdu, transmit mpdu, lost mpdu, number of retries)
	[COMMENT] : WME Background Access Category (receive mpdu, transmit mpdu, lost mpdu, number of retries)
	[COMMENT] : WME Video Access Category (receive mpdu, transmit mpdu, lost mpdu, number of retries)
	[COMMENT] : WME Voice Access Category (receive mpdu, transmit mpdu, lost mpdu, number of retries)

** [CLUSTER] : 33
	[COMMENT] : Encode a CA certificate alias so it does not contain illegal character. @hide /
	[COMMENT] : Set CA certificate alias. See the {@link android.security.KeyChain} for details on installing or choosing a certificate @param alias identifies the certificate @hide /
	[COMMENT] : Set CA certificate aliases. When creating installing the corresponding certificate to the keystore, please use alias encoded by {@link #encodeCaCertificateAlias(String)}. See the {@link android.security.KeyChain} for details on installing or choosing a certificate. @param aliases identifies the certificate @hide /
	[COMMENT] : Get CA certificate alias @return alias to the CA certificate @hide /
	[COMMENT] : Get CA certificate aliases @return alias to the CA certificate @hide /
	[COMMENT] : Set Client certificate alias. See the {@link android.security.KeyChain} for details on installing or choosing a certificate @param alias identifies the certificate @hide /
	[COMMENT] : Get client certificate alias @return alias to the client certificate @hide /

** [CLUSTER] : 115
	[COMMENT] : Represents the notification parameters that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /
	[COMMENT] : Represents the Android-specific options that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /
	[COMMENT] : Represents the Webpush-specific notification options that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /
	[COMMENT] : Represents the Webpush protocol options that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /
	[COMMENT] : Represents the Android-specific notification options that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /

** [CLUSTER] : 116
	[COMMENT] : Creates a new {@code Notification} using the given title and body. @param title Title of the notification. @param body Body of the notification. /
	[COMMENT] : Creates a new notification with the given title and body. Overrides the options set via {@link Notification}. @param title Title of the notification. @param body Body of the notification. /
	[COMMENT] : Creates a new notification with the given title, body and icon. Overrides the options set via {@link Notification}. @param title Title of the notification. @param body Body of the notification. @param icon URL to the notifications icon. /

** [CLUSTER] : 117
	[COMMENT] : Sets the title of the alert. When provided, overrides the title sent via {@link Notification}. @param title Title of the notification. @return This builder. /
	[COMMENT] : Sets the body of the alert. When provided, overrides the body sent via {@link Notification}. @param body Body of the notification. @return This builder. /
	[COMMENT] : Sets the title of the Android notification. When provided, overrides the title set via {@link Notification}. @param title Title of the notification. @return This builder. /
	[COMMENT] : Sets the body of the Android notification. When provided, overrides the body sent via {@link Notification}. @param body Body of the notification. @return This builder. /

** [CLUSTER] : 118
	[COMMENT] : Sets the key of the text in the app's string resources to use to localize the action button text. @param actionLocKey Resource key string. @return This builder. /
	[COMMENT] : Sets the key of the body string in the app's string resources to use to localize the body text. @param locKey Resource key string. @return This builder. /
	[COMMENT] : Sets the key of the title string in the app's string resources to use to localize the title text. @param titleLocKey Resource key string. @return This builder. /
	[COMMENT] : Sets the key of the body string in the app's string resources to use to localize the body text. @param bodyLocKey Resource key string. @return This builder. /

** [CLUSTER] : 119
	[COMMENT] : Adds a resource key string that will be used in place of the format specifiers in {@code bodyLocKey}. @param arg Resource key string. @return This builder. /
	[COMMENT] : Adds a list of resource keys that will be used in place of the format specifiers in {@code bodyLocKey}. @param args List of resource key strings. @return This builder. /
	[COMMENT] : Adds a resource key string that will be used in place of the format specifiers in {@code titleLocKey}. @param arg Resource key string. @return This builder. /
	[COMMENT] : Adds a list of resource keys that will be used in place of the format specifiers in {@code titleLocKey}. @param args List of resource key strings. @return This builder. /

** [CLUSTER] : 120
	[COMMENT] : Retrieves an {@link android.mtp.MtpDevice} object for the USB device with the given name. @param deviceName the name of the USB device @return the MtpDevice, or null if it does not exist /
	[COMMENT] : Retrieves an {@link android.mtp.MtpDevice} object for the USB device with the given ID. @param id the ID of the USB device @return the MtpDevice, or null if it does not exist /
	[COMMENT] : Retrieves a list of all currently connected {@link android.mtp.MtpDevice}. @return the list of MtpDevices /

** [CLUSTER] : 121
	[COMMENT] : Retrieves a list of all {@link android.mtp.MtpStorageInfo} for the MTP or PTP device with the given USB device name @param deviceName the name of the USB device @return the list of MtpStorageInfo /
	[COMMENT] : Retrieves the {@link android.mtp.MtpObjectInfo} for an object on the MTP or PTP device with the given USB device name with the given object handle @param deviceName the name of the USB device @param objectHandle handle of the object to query @return the MtpObjectInfo /
	[COMMENT] : Deletes an object on the MTP or PTP device with the given USB device name. @param deviceName the name of the USB device @param objectHandle handle of the object to delete @return true if the deletion succeeds /
	[COMMENT] : Retrieves a list of {@link android.mtp.MtpObjectInfo} for all objects on the MTP or PTP device with the given USB device name and given storage ID and/or object handle. If the object handle is zero, then all objects in the root of the storage unit will be returned. Otherwise, all immediate children of the object will be returned. If the storage ID is also zero, then all objects on all storage units will be returned. @param deviceName the name of the USB device @param storageId the ID of the storage unit to query, or zero for all @param objectHandle the handle of the parent object to query, or zero for the storage root @return the list of MtpObjectInfo /

** [CLUSTER] : 163
	[COMMENT] : Returns the list of object handles for all objects on the given storage unit, with the given format and parent. Information about each object can be accessed via {@link #getObjectInfo}. @param storageId the storage unit to query @param format the format of the object to return, or zero for all formats @param objectHandle the parent object to query, -1 for the storage root, or zero for all objects @return the object handles, or null if fetching object handles fails /
	[COMMENT] : Retrieves the object handle for the parent of an object on the device. @param objectHandle handle of the object to query @return the parent's handle, or zero if it is in the root of the storage /
	[COMMENT] : Retrieves the ID of the storage unit containing the given object on the device. @param objectHandle handle of the object to query @return the object's storage unit ID /
	[COMMENT] : Returns the object handle for the object's parent Will be zero for the root directory of a storage unit @return the object's parent /

** [CLUSTER] : 165
	[COMMENT] : This class encapsulates information about a storage unit on an MTP device. This corresponds to the StorageInfo Dataset described in section 5.2.2 of the MTP specification. /
	[COMMENT] : This class encapsulates information about an object on an MTP device. This corresponds to the ObjectInfo Dataset described in section 5.3.1 of the MTP specification. /
	[COMMENT] : This class encapsulates information about an MTP device. This corresponds to the DeviceInfo Dataset described in section 5.1.1 of the MTP specification. /

** [CLUSTER] : 166
	[COMMENT] : Returns the storage ID for the storage unit. The storage ID uniquely identifies the storage unit on the MTP device. @return the storage ID /
	[COMMENT] : Returns the description string for the storage unit. This is typically displayed to the user in the user interface on the MTP host. @return the storage unit description /
	[COMMENT] : Returns the volume identifier for the storage unit @return the storage volume identifier /
	[COMMENT] : Returns the object handle for the MTP object @return the object handle /
	[COMMENT] : Returns the storage ID for the MTP object's storage unit @return the storage ID /
	[COMMENT] : Returns the format code for the MTP object @return the format code /
	[COMMENT] : Returns the size of the MTP object @return the object size /
	[COMMENT] : Returns the format code for the MTP object's thumbnail Will be zero for objects with no thumbnail @return the thumbnail format code /
	[COMMENT] : Returns the size of the MTP object's thumbnail Will be zero for objects with no thumbnail @return the thumbnail size /
	[COMMENT] : Returns the width of the MTP object's thumbnail in pixels Will be zero for objects with no thumbnail @return the thumbnail width /
	[COMMENT] : Returns the height of the MTP object's thumbnail in pixels Will be zero for objects with no thumbnail @return the thumbnail height /
	[COMMENT] : Returns the width of the MTP object in pixels Will be zero for non-image objects @return the image width /
	[COMMENT] : Returns the height of the MTP object in pixels Will be zero for non-image objects @return the image height /
	[COMMENT] : Returns the depth of the MTP object in bits per pixel Will be zero for non-image objects @return the image depth /
	[COMMENT] : Returns the name of the MTP object @return the object's name /
	[COMMENT] : Returns the storage ID for the storage unit @return the storage ID /
	[COMMENT] : Returns the file path for the storage unit's storage in the file system @return the storage file path /
	[COMMENT] : Returns the description string for the storage unit @return the storage unit description /

** [CLUSTER] : 167
	[COMMENT] : Object is not protected. It may be modified and deleted, and its properties may be modified. /
	[COMMENT] : Object can not be modified or deleted and its properties can not be modified. /
	[COMMENT] : Object can not be modified or deleted but its properties are modifiable. /
	[COMMENT] : Object's contents can not be transfered from the device, but the object may be moved or deleted and its properties may be modified. /

** [CLUSTER] : 168
	[COMMENT] : The {@link Intent} that must be declared as handled by the service. /
	[COMMENT] : The {@link Intent} that must be declared as handled by the service. The service must also require the {@link android.Manifest.permission#BIND_TV_REMOTE_SERVICE} permission so that other applications cannot abuse it. /
	[COMMENT] : The {@link Intent} that must be declared as handled by the service. Put this in your manifest. /