Software Design Specifications

For

Stylo: Stylometry Tool

Prepared By:
Kyle Musal
Aaron Chapin
Andrew Orner
Matthew Tornetta

Advised By:
Rachel Greenstadt
Jeff Salvage

Drexel University

Table of Contents

1. Document History

Version	Date	Editor	Description
1.0	1/28/2010	MT, AC, KM, AO	Initial Version

2. Introduction

2.1. Purpose

The purpose of this document is to describe, in detail, the design and architecture of Stylo (henceforth referred to as “the software”) in its entirety. These design decisions pertain to the functionality, performance, and external interfaces of the software. This document will be used to implement all features as described in the requirements document for the software.

2.2. Scope

This document pertains to the design and architecture of the first release of Stylo and includes all functional requirements as listed in the requirements document for the software. This document is intended for the developers and testers of the software.

2.3. Design Goals

The software is designed with the following goals in mind:

- **Accuracy:** The software’s analysis is as accurate as existing tools, within reason.
- **Ease of Use:** The software interface is intuitive and is simple to use by anyone, regardless of familiarity with stylometry.
- **Modularity**: Through its plug-in system, the software supports interchanging graphical toolkits, machine learning tools, and available feature sets used for analysis.
- **Open Source:** The software is open source and licensed under the GNU General Public License.
- **Portability:** The software can run on a Windows, Linux, or Mac platform without any loss of functionality or stability.
- **Security:** Due to the possible sensitivity of works being analysed, the software does not knowingly transmit any information over the network and stores all data in an encrypted format.

2.4. Glossary

Bigram: A 2-gram
Corpus (pl. Corpora): A large, structured set of texts from various authors.
Feature: The smallest unit of criteria the software can analyse about a document.
Feature Set: A group of one or more similar features.
Graphical Toolkit: A set of tools for creating a graphical interface for a software.

Machine Learning Tool: A program that can change its behaviour based on sample data; for example a program that can learn how to recognize patterns or features.

N-Gram: A sub-sequence of *n* items from a given sequence. For example, the sentence “I painted the fence red.” has the 2-grams “I painted”, “painted the”, “the fence”, “fence red”.

Plug-in: A modular extension to the software that adds support for additional functionality of the software.

Trigram: A 3-gram

Stylometry: The study of author identification through analysis of linguistic style.

Unigram: A 1-gram

2.5. References

Anonymouth: < http://www.cs.drexel.edu/~pv42/thebiz/ >

NLTK: < http://www.nltk.org/ >

NodeBox Linguistics Library: < http://nodebox.net/code/index.php/Linguistics >

OleFileIO_PL: < http://www.decalage.info/python/olefileio >

Orange: < http://orange.biolab.si/ >

PyCrypto: < http://www.dlitz.net/software/pycrypto/ >

pyPdf: < http://pybrary.net/pyPdf/ >

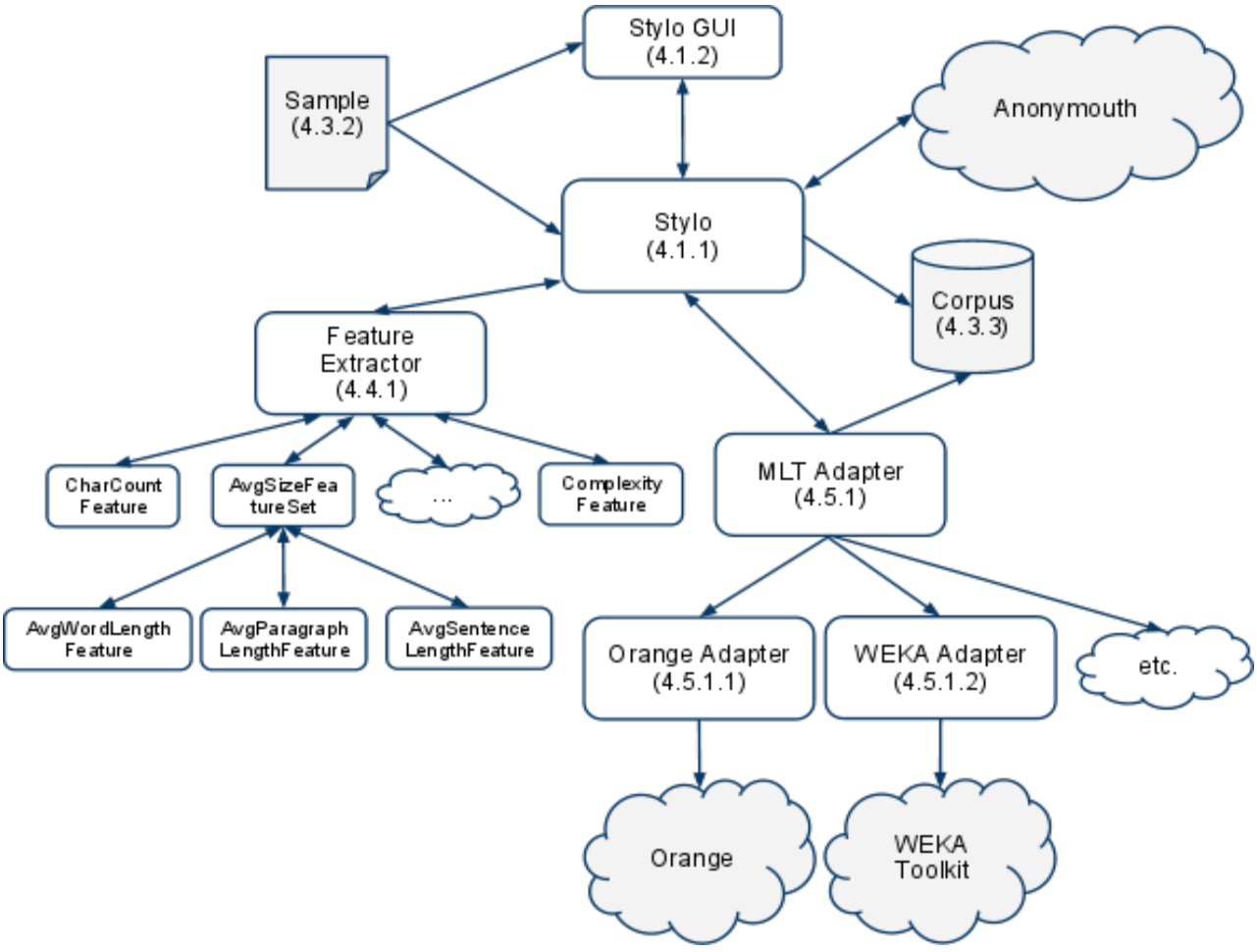
PyRTF: < http://pyrtf.sourceforge.net/ >

Python: < http://www.python.org/ >

TkInter: < http://wiki.python.org/moin/TkInter >

WordNet: < http://wordnet.princeton.edu/ >

2.6. Context Diagram



3. Architecture

3.1. Technologies Used

The software is completely written in Python. By default, the software uses TkInter, the graphical toolkit that is packaged with Python, for its graphical interface. The software also uses Orange as its default machine learning interface.

3.1.1. Python 2.6

Python was chosen as the language of choice for the software because of its ability to be used on a variety of operating systems. It was also chosen because its dynamic nature allows Stylo to have a plug-in system that does not require recompiling source code. Python's simplicity will also enable users with programming knowledge to extend the software with new linguistic features with little hassle. Python 2.6 is specifically targeted as it is the most standard Python version across many Unix-Like operating systems.

3.1.2. TkInter

TkInter will be the default GUI framework that the software uses to expose its functionality to the end user. TkInter was chosen in large part to maintain a similar look and feel to the Anonymouth project which also will be using TkInter. In addition TkInter is compact and distributed with Python which will reduce dependencies in some cases.

3.1.3. Orange

Orange was chosen as the machine learning tool for the software because of its native support for Python. Orange has a strong developer and user base which ensures its active development.

3.1.4. (NLTK) Natural Language Tool Kit

Natural language processing will play a large part in the work the software will be doing. As such we needed a feature rich tool kit to use to accomplish the task of processing the natural language contained in the samples. NLTK's native support for Python and large feature list made it the logical choice.

3.1.5. NodeBox Linguistics Library

Some linguistic features require a database of words to compare against or look up synonyms for. WordNet was the first option we looked into but NodeBox packages the WordNet database up into a library with some other useful features so we decided to go with NodeBox.

3.1.6. PyCrypto

The samples included in each corpus as well as the information generated by Stylo contain potentially sensitive information. To protect this information we are including the option to encrypt all information stored by Stylo (samples, feature results, authorship results, etc). PyCrypto is the most fully featured cryptography library for Python and as such will be the library we're using.

3.1.7. Text Extraction Tools

Various libraries are needed to access text contain in file types other than plain text. Each tool that was chosen (pyPdf, OleFileIO_pl, PyRTF) because it was decided they were the most fully featured, cross platform, and up to date libraries of their class.

3.2. Application Overview

Upon completion, the software will be able to analyse features from selected texts. The extracted features will then be sent through a machine learning tool and compared against a corpus (which has previously been trained) in an attempt to attribute ownership of texts to an author contained in the selected corpus. In addition, the software will be able to communicate with the Anonymouth software to assist the end user with anonymizing his or her writing.

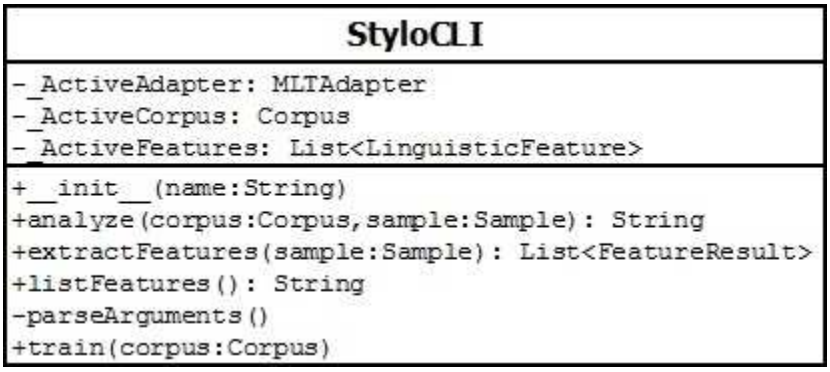
4. Detailed Design

Unless mentioned otherwise, all classes contain getter and setter methods for private member variables.

4.1. Program Interface

4.1.1. Command Line Interface

The main Stylo class is the entry point of the software when run through the command line. This class loads the selected texts, delegates feature extraction and machine learning analysis to the appropriate subsystems, and returns with the results.



Properties

ActiveCorpus	The corpus that is currently active, that will be trained or used when analyzing a document.
ActiveAdapter	The active MLT Adapter that will be used when training or analyzing a document.
ActiveFeatures	The list of active features that will be used for extraction.

Methods

analyze	Runs the analyze function on a specific document. Returns a string giving details of the analysis. Can either be a status message (“The operation completed successfully”) or, if set by the user, the full analysis documentation.
extractFeatures	Extract the active features from the specified sample.
listFeatures	Prints out a help message listing what features are available for use.

parseArguments	Parse the command line arguments to set up the program.
train	Trains the default corpus, or a non-default corpus if specified.

4.1.1.1. Command Line Flags

4.1.1.1.1. -c: Corpus

Indicates the path to the corpus to use, if not the current one.

4.1.1.1.2. -f: Features

A semicolon delimited list follows this flag, detailing which features to analyze. If this flag is not given, all available feature sets are used.

4.1.1.1.3. -ff: Feature File

A path to a text file containing a line-delimited list of features to use for analysis.

4.1.1.1.4. -i: Input

The path to the document, or folder containing the documents, to be analysed.

4.1.1.1.5. -l: List

Outputs a list of the features that can be utilized, for easy reference.

4.1.1.1.6. -m: Machine Learning Tool

Selects the machine learning tool to use for training and analysis.

4.1.1.1.7. -o: Output

Allows the user to specify the path where the analysis results are stored.

4.1.1.1.8. -p: Print

Specifies that output is to be human-readable, instead of serialized.

4.1.1.1.9. -t: Train

Tells the software to only train on the current corpus, rather than analyzing a document.

4.1.1.1.10. -d: Training Data Location

Specifies the path where the training data is stored.

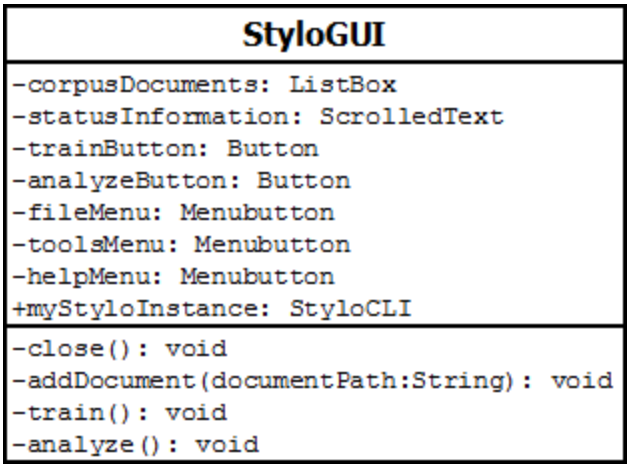
4.1.1.1.11. -h: Help

Displays information about the software and prints out all available command line flags.

4.1.2. Graphical User Interface

The StyloGUI class is the entry point of the software when run as a graphical user interface. This class acts as a wrapper over the command line interface. The user is presented

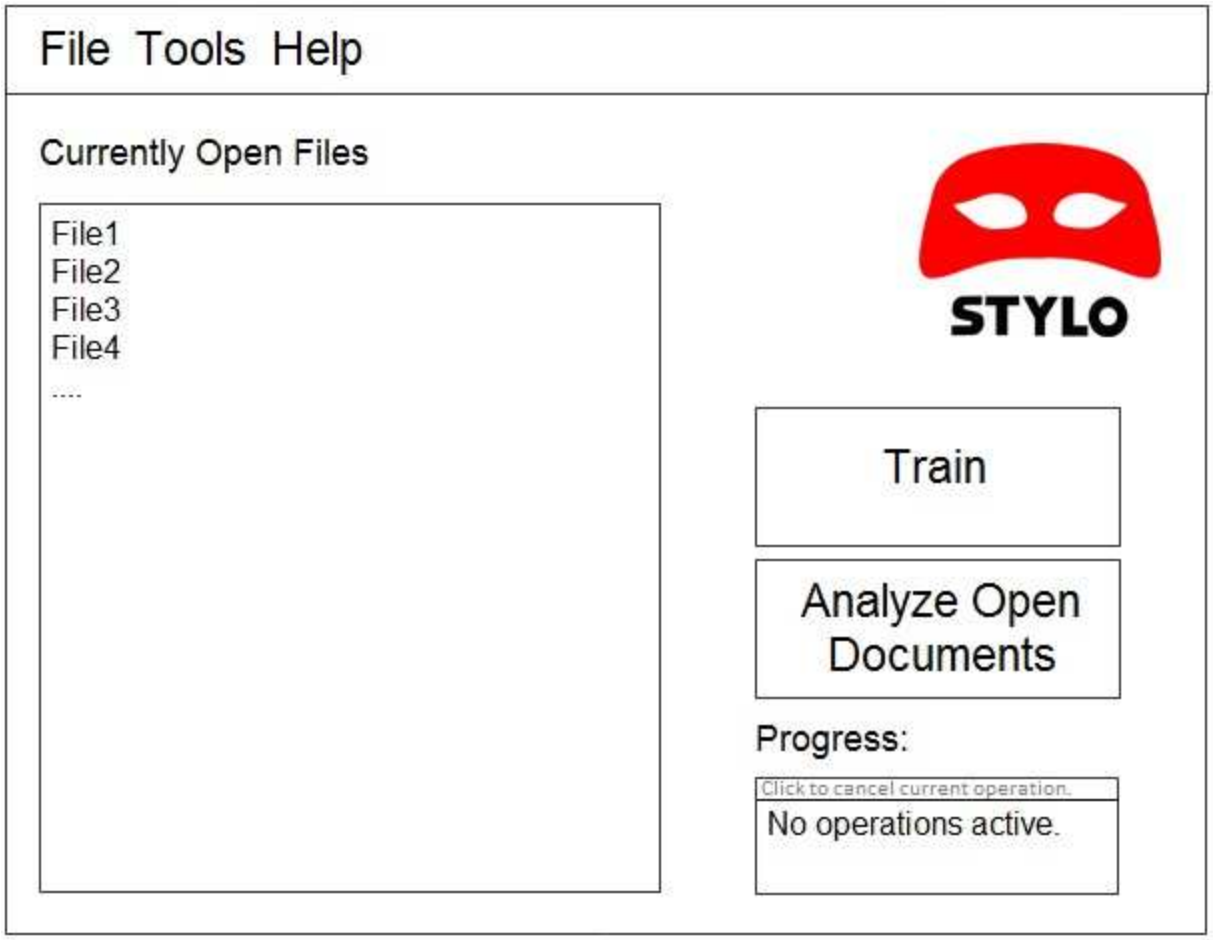
with menus and buttons to configure the parameters of the software, and then passes the appropriate commands to the command line interface. Any output received from the command line interface is displayed to the user.



Properties

activeDocuments	A list showing the documents pending analysis.
statusInformation	A text box that is used to print out information to the user. A bar will appear over this tool box during analysis and training that will let the user cancel the current operation.
trainButton	A button that, when pressed, calls the train() function.
analyzeButton	A button that, when pressed, calls the analyze() function.
fileMenu	A collection of menu options, such as “Add Document” and “Exit”
toolsMenu	A collection of menu options regarding operation of the program, such as “Manage Features” and “Configure Machine Learning Tool”
myStyloInstance	An instance of the CLI main class, that performs the required tasks.
close	Closes the GUI, cleans up any allocated resources, and exits
addDocument	Adds a document to the current corpus.
train	Signals the CLI to train on the current corpus. While training is occurring, this button becomes disabled.
analyze	Signals the CLI to begin an analysis procedure. While analysis is occurring, this button becomes disabled. If analysis is expected to take longer than two minutes, the user is given an option to cancel operation right away.

4.1.2.1. Mockup



4.1.2.2. File Menu

4.1.2.2.1. Open Document

Allows the user to browse for and add a document to the list of documents pending analysis.

4.1.2.2.2. Exit

Exits the software.

4.1.2.3. Tools Menu

4.1.2.3.1. Select Corpora

Allows the user to select a corpora for use in analysis.

4.1.2.3.2. Select Features

Allows the user to select features used for analysis. Selected features will be stored in a file contained inside of the ‘Stylo’ folder in order to preserve selected features between

sessions. By default all available features are selected.

4.1.2.3.3. Manage Plug-Ins

Allows the user to enable or disable active plug-ins.

4.1.2.3.4. Manage Machine Learning Tools

Allows the user to configure installed machine learning tool adapters. This window allows the user to select the active machine learning tool, and allows the user to configure each installed machine learning tool based on its list of configuration options.

4.1.2.4. Help Menu

4.1.2.4.1. Help

Opens a window containing documentation for the software to assist the user in using the program.

4.1.2.4.2. About

Opens a dialog box containing information about the program, any required license information, etc.

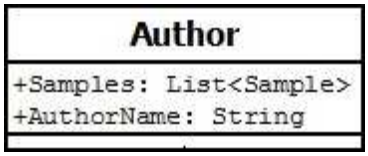
4.1.3. Default Run Action

If the user does not specify any flags when starting the software, the software runs in graphical mode.

4.3. Business Logic

4.3.1. Author

The Author class represents an author that is part of a selected corpus.

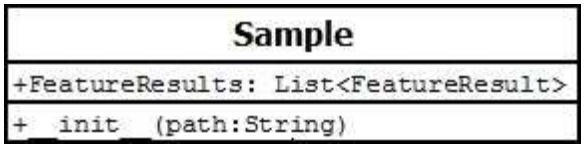


Properties

Samples	List of samples in the selected corpus belonging to this author.
AuthorName	Name of this author.

4.3.2. Sample

The Sample class represents a writing sample on which the user wants to perform analysis

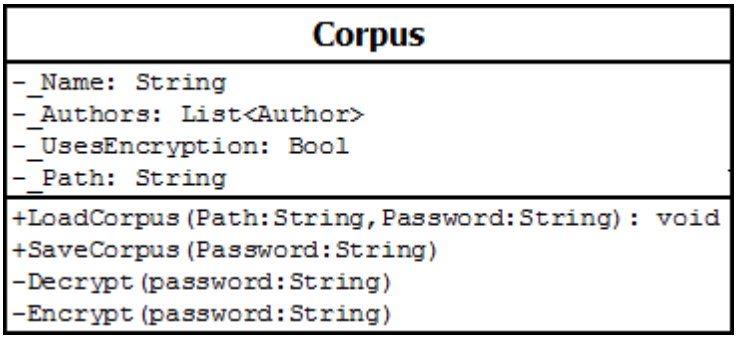


Properties

FeatureResults	List of FeatureResults describing this sample. This list is initially empty, and is only filled upon analysis.

4.3.3. Corpus

The Corpus class represents a corpus of authors.



Properties

Name	Name of the corpus.
Authors	List of authors contained in this corpus.
UsesEncryption	Marks whether this corpus should be stored with encryption.
Path	Path that this corpus was stored at.

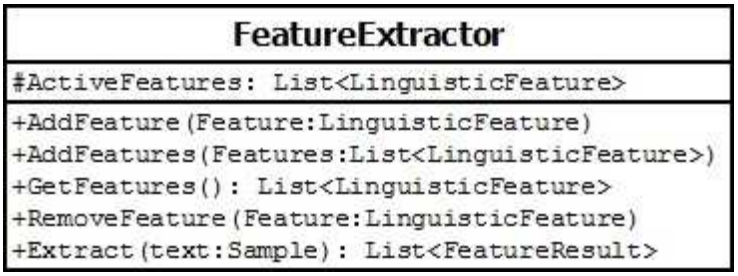
Methods

LoadCorpus	Load a corpus from disk, potentially with an encryption password.
SaveCorpus	Saves corpus to disk with the supplied password.
Decrypt	Decrypts the corpus using the supplied password.
Encrypt	Encrypts the corpus using the supplied password.

4.4. Feature Extraction

4.4.1. FeatureExtractor

The FeatureExtractor class is responsible for starting the feature extraction process on all selected features.



Properties

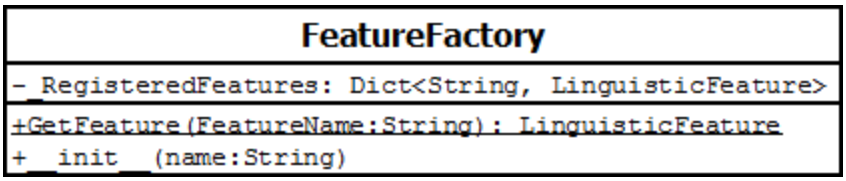
ActiveFeatures	List of features that are selected for extraction.

Methods

AddFeature	**Input:** LinguisticFeature to be added to list of active features. **Logic:** Add feature to list of active features.
AddFeatures	**Input:** List of LinguisticFeatures to be added to the list of active features. **Logic:** Add all LinguisticFeatures from input list to list of active features.
GetFeatures	**Output:** Returns list of active features.
RemoveFeature	**Input:** LinguisticFeature to remove from list of active features. **Logic:** Removes input LinguisticFeature from list of active features. If list of active features does not contain input LinguisticFeature, nothing is done.
Extract	**Input:** Sample to perform feature extraction on. **Logic:** Perform feature extraction on input sample using all active features. **Output:** Collects and returns all FeatureResults that active features returned from extraction.

4.4.2. FeatureFactory

The FeatureFactory class acts as a single point from which the software can obtain any LinguisticFeature currently installed using only its name. Because features are installed as Python modules, there can be no conflict between feature names.

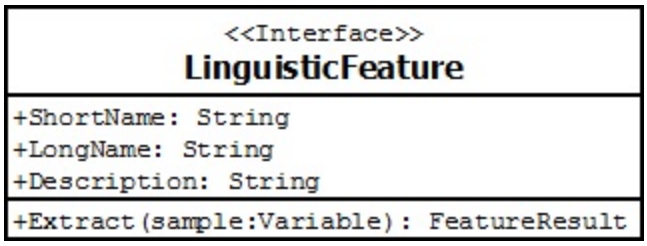


Methods

GetFeature	**Input:** Short name of desired feature **Logic:** Finds installed LinguisticFeature with the given short name. **Output:** Returns the LinguisticFeature with the given short name.

4.4.3. LinguisticFeature

The LinguisticFeature interface is a common interface that all features and feature sets derive from.



Properties

Short Name	A shortened version of the feature name. This is the name by which the feature is referenced by command line.
Long Name	The full name of the feature. This is used for display purposes.
Description	A short description of the feature. This is used for display purposes.

Methods

Extract	**Input:** The sample being analysed. **Output:** FeatureResult describing results of analysis of this feature.

4.4.4. Implemented Features

The software comes packaged with the following feature sets.

4.4.4.1. Character Count

Short Name	CharCount

Long Name	Character Count
Description	Number of characters in the sample.
Input Type	Plain Text
Return Type	Integer
Logic	Return a simple count of all characters in a sample.

4.4.4.2. Word Count

Short Name	WordCount
Long Name	Word Count
Description	Number of words in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Call len() on NLTK text returns the number of words.

4.4.4.3. Large Word Count

Short Name	LargeWordCount
Long Name	Large Word Count
Description	Number of words in the sample with a length greater than 10.
Input Type	NLTK Text
Return Type	Integer
Logic	Returns a count of the number of words in a sample with a length greater than 10 letters.

4.4.4.4. Unique Word Count

Short Name	UniqueWordCount
Long Name	Unique Word Count
Description	Number of unique words in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Call len() on NLTK text returns the number of words.

4.4.4.5. Punctuation Count

Short Name	PunctCount
Long Name	Punctuation Count
Description	Number of each type of punctuation in the sample.
Input Type	Plain Text
Return Type	List<Integer>
Logic	Count the number of occurrences of each of the types of punctuation.

4.4.4.6. Sentence Count

Short Name	SentenceCount
Long Name	Sentence Count
Description	Number of sentences in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Count the number of sentences in the sample.

4.4.4.7. Percent Large Words

Short Name	PctLargeWords
Long Name	Percent of Large Words
Description	Percentage of large words in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Divide number of large words in sample by number of words in the sample.

4.4.4.8. Percent Punctuation

Short Name	PctPunctuation
Long Name	Percentage of Punctuation
Description	Percentage of each punctuation in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Divide the number of each punctuation by the number of words in the sample.

4.4.4.9. Percent Word Group Words

Short Name	PctWordGroups
Long Name	Percentage of Word Group Words
Description	Percentage of words in the sample that belong to each word group
Input Type	NLTK Text
Return Type	Integer

Logic	Divide total number of words in each word group by the number of words in the sample. See **Appendix A2** for more information on word groups.

4.4.4.10. Average Sentence Length

Short Name	AvgSentenceLength
Long Name	Average Sentence Length
Description	Average number of words per sentence.
Input Type	NLTK Text
Return Type	Integer
Logic	Divide total number of words in the sample by the number of sentences in the sample.

4.4.4.11. Average Syllables per Word

Short Name	AvgSyllablesPerWord
Long Name	Average Number of Syllables per Word
Description	Average number of syllables per word in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Divide the total number of syllables in the sample by the word count of the sample.

4.4.4.12. Average Word Length

Short Name	AvgWordLength
Long Name	Average Word Length

Description	Average length of words in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Number of alpha-numeric characters divided by number of words.

4.4.4.13. Average Characters Per Paragraph

Short Name	AvgCharactersPerParagraph
Long Name	Average Number of Characters per Paragraph
Description	Average number of characters per paragraph in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Divide total characters in sample by number of paragraphs in sample.

4.4.4.14. Average Words Per Paragraph

Short Name	AvgWordsPerParagraph
Long Name	Average Number of Words per Paragraph
Description	Average number of words per paragraph in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Divide total number of words in sample by number of paragraphs in sample.

4.4.4.15. Average Sentences Per Paragraph

Short Name	AvgSentencesPerParagraph
Long Name	Average Number of Sentences per Paragraph
Description	Average number of sentences per paragraph in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Divide total number of sentences in sample by number of paragraphs in sample.

4.4.4.16. Word Groups

Short Name	WordGroups
Long Name	Number of Words per Word Group
Description	Number of words for each defined word group.
Input Type	NLTK Text
Return Type	Integer
Logic	Compute the total number of words in each registered word group. See **Appendix A2** for more information on word groups.

4.4.4.17. Sample Complexity

Short Name	Complexity
Long Name	Sample Complexity
Description	Percentage of unique words in the sample.
Input Type	NLTK Text
Return Type	Integer
Logic	Divide the number of unique words by the total number of words, resulting in a value between 0 and 1.

4.4.4.18. Sample Readability

Short Name	Readability
Long Name	Sample Readability
Description	Comparison between the size of the document and the percentage of words with more than 3 syllables.
Input Type	NLTK Text
Return Type	Integer
Logic	0.4 * (AvgSentenceLength + 100 * NumberWordsWithMoreThan3Syllables / WordCount)

4.4.4.19. Synonym Classification

Short Name	SynClassification
Long Name	Synonym Classification
Description	Author's choice of words compared to possible synonyms.
Input Type	NLTK Text
Return Type	Double
Logic	Sum of the products of the number of synonyms for each word and the minimum number of times that word appears in sample and corpus text.

4.4.4.20. N-Gram Frequency

Short Name	NGramFrequency
Long Name	N-Gram Frequency
Description	The frequency of which all n-grams appear in the sample. By default, this uses unigrams, bigrams, and trigrams.

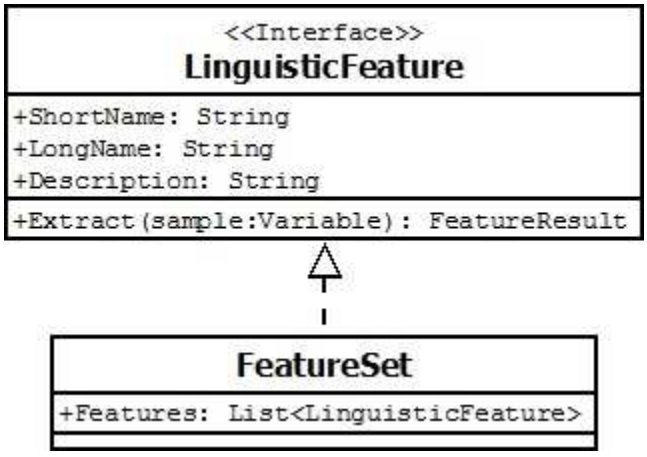
Input Type	NLTK Text
Return Type	Double
Logic	For each unique n-gram in the sample, divide the number of times this n-gram appears by the total number of n-grams of the same size in the document.

4.4.4.21. Lexical Diversity

Short Name	LexicalDiversity
Long Name	Lexical Diversity
Description	Measures the diversity of the words in the sample from 0-1; 0 being no diversity and 1 being infinite diversity.
Input Type	NLTK Text
Return Type	Double
Logic	Given n : frequency of a word in the sample and N : total number of words Diversity = 1 - (sum(n * (n-1)) / (N * (N-1)))

4.4.5. FeatureSet

The FeatureSet class contains multiple LinguisticFeatures within itself, and proxies interface calls to all contained Features.

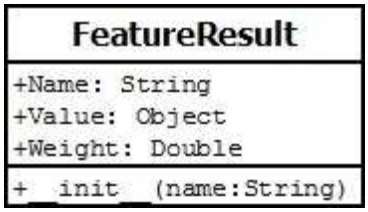


Properties

Features	List of features that this FeatureSet will proxy calls through to.

4.4.6. FeatureResult

Once a LinguisticFeature extracts information from a sample, it stores its result in a FeatureResult object.



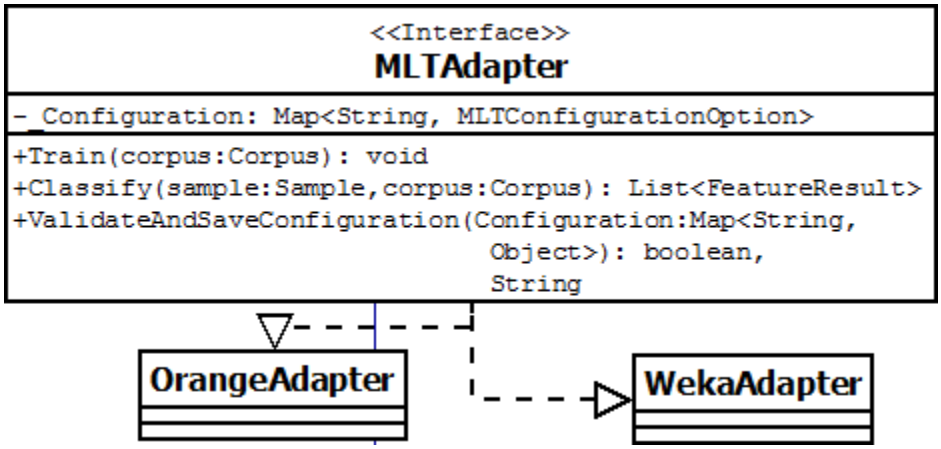
Properties

Name	Name of the LinguisticFeature associated with this FeatureResult.
Value	Value that the associated LinguisticFeature extracted from the sample.
Weight	Relative weight of this feature used in attribution of authorship.

4.5. Machine Learning Tool Interface

4.5.1. MLTAdapter

The MLTAdapter interface will allow adaptation to installed machine learning tools. This interface does not have a setter for Configuration.



Properties

Configuration	Key-Value pair of configuration options. Example options include type of learning method to use, parameters for analysis, etc. This will be specific to each adapter.

Methods

Train	**Input:** Corpus to train on **Logic:** Trains the machine learning tool against the input corpus.
Classify	**Input:** Sample to classify, corpus to classify against **Logic:** If the corpus has not yet been trained, train the corpus first. Attempt to attribute an author from the input corpus to the input sample. **Output:** A list of FeatureResult objects populated based on the result of classification. The list will contain a special FeatureResult object containing the predicted author as the Value and predicted accuracy as the Weight.
ValidateAndSave Configuration	**Input:** Key-Value pairs of configuration changes **Output**: True if all values are valid and configuration was saved, or False if configuration was invalid. If configuration was invalid, also returns an error message.

4.5.1.1. OrangeAdapter

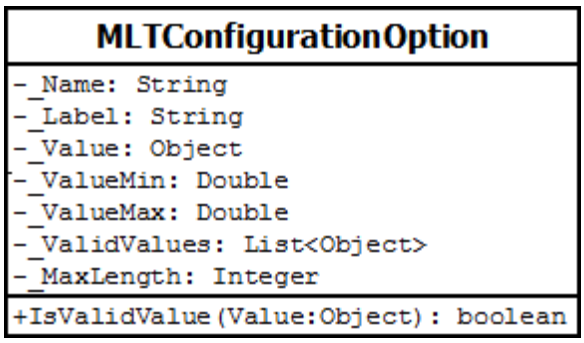
The OrangeAdapter class interfaces specifically with the Orange machine learning toolkit.

4.5.1.2. WekaAdapter

The WekaAdapter class interfaces specifically with the WEKA machine learning toolkit.

4.5.2. MLTConfigurationOption

The MLTConfigurationOption represents a single configuration option for a machine learning tool. The specific adapters are responsible for interpreting the value of the option.



Properties

Name	Name of this configuration option.
Label	Label of this configuration option, to be used for the GUI configuration window.
Value	Value of this configuration option.
ValueMin	Minimum value that this option can contain. Optional.
ValueMax	Maximum value that this option can contain. Optional.
ValidValues	List of valid values that this option can contain. Optional.
MaxLength	Maximum length of this option's value. Optional.

Methods

IsValidValue	Determines whether the passed in value would be valid.

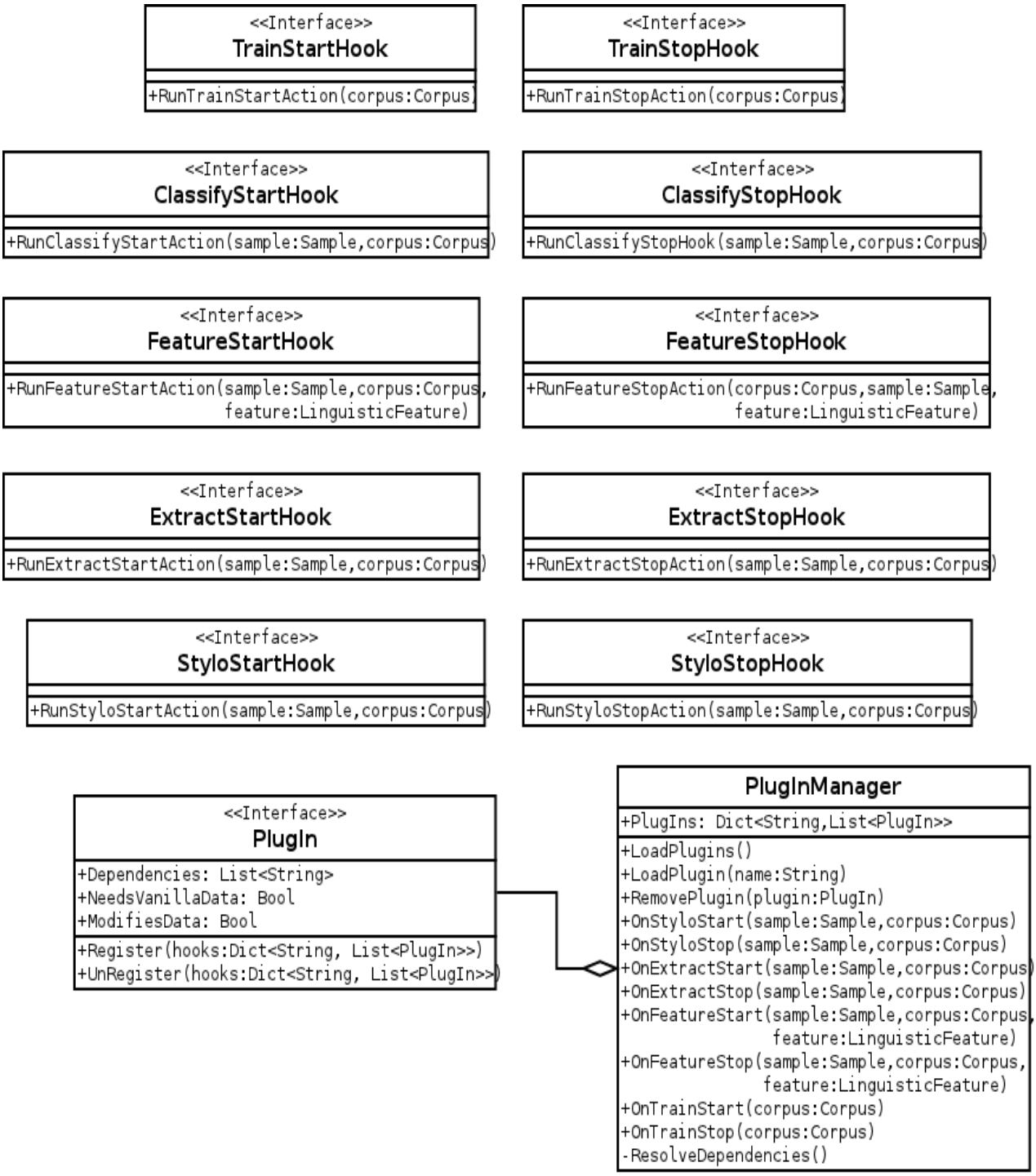
4.6. Plug-In Architecture

4.6.1. Overview

A plug-in system is being designed into Stylo so that in the future, developers external to the original team can hook into core functionality with minimal modifications to the code. The plug-in system would allow for additional, modular features to be added or removed from Stylo without the need to modify code in other areas.

One of the major issues in the design of the plug-in system is deciding in which order plug-ins execute when they need to work on the same information. To help alleviate this competition, Stylo plug-ins will contain a few variables that describe their functionality. We will then use these variables to decide on what order to execute plug-ins should there be multiple plug-ins wanting to execute on the same information.

4.6.2. Hook Interfaces



4.6.3. PlugIn Object

4.6.3.1. Function and Variables

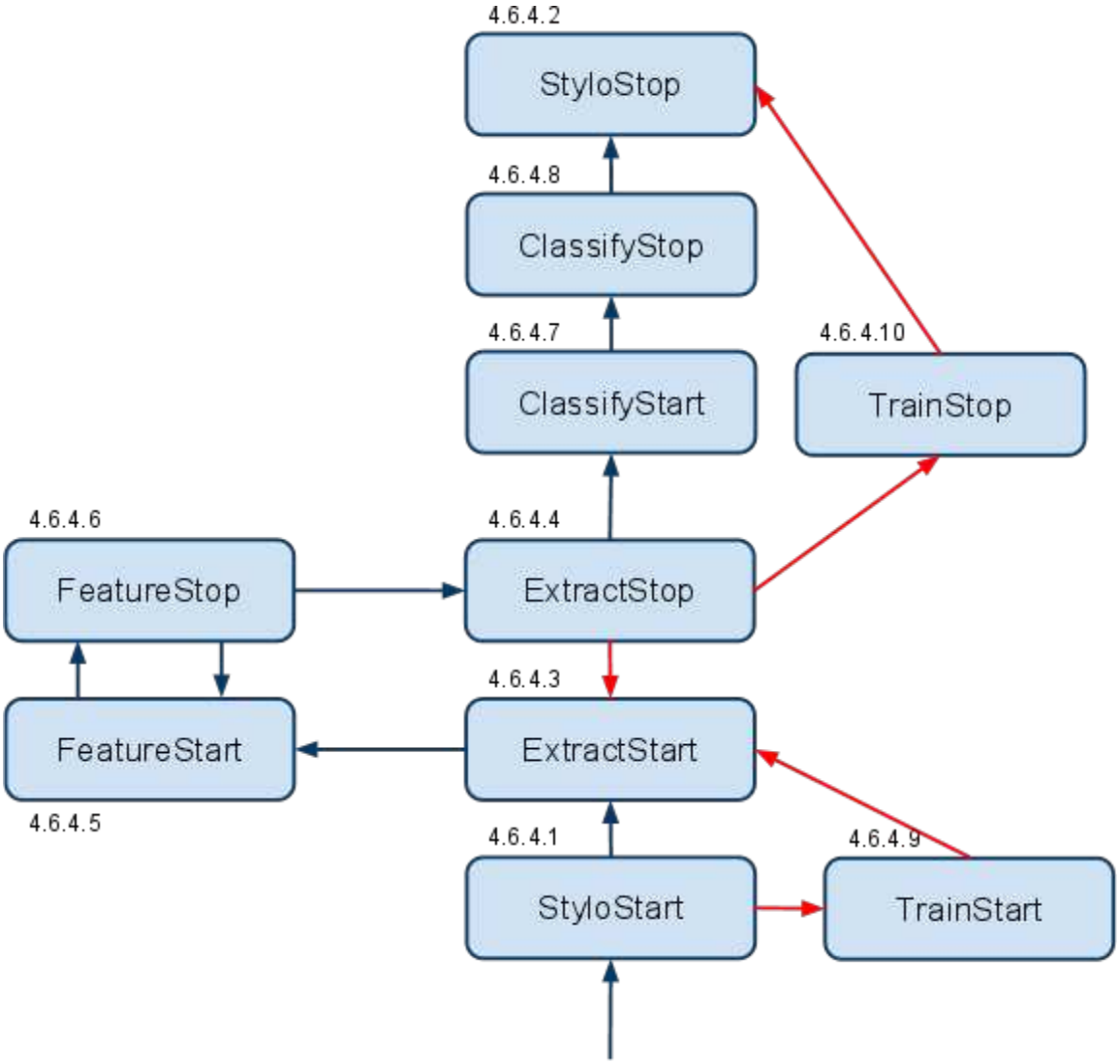
Dependencies	List of names of other plug-ins that have to run before current one.

NeedsVanillaData	Indicates this plug-in needs unmodified data.
ModifiesData	Indicates that this plug-in will modify the data it is given.
Register	Register a plug-in to receive notification of various events.
UnRegister	Remove a plug-in from all events.

4.6.3.2. Description

The PlugIn object is the heart of Stylo and is contained inside a Python package. All functionality outside of loading and registering plug-ins will be implemented as a plug-in. Plug-ins must implement one or more of the various interfaces that define a hook. These interfaces define the method that will be called when that event takes place and if not present, the plug-in will fail to be executed. Since Python supports multiple inheritance, each plug-in can hook into any number of the available events.

4.6.4. Available Hooks



4.6.4.1. StyloStart

When	Immediately after Stylo starts.
Uses	Initialization of plug in that needs access to corpus or sample.

4.6.4.2. StyloEnd

When	Before Stylo exits.
Uses	Clean up of plug in data before Stylo exits.

4.6.4.3. ExtractStart

When	Before feature extraction begins.
Uses	Modify sample before doing extraction.

4.6.4.4. ExtractEnd

When	At the end of feature extraction
Uses	Access feature data before classification is attempted.

4.6.4.5. FeatureStart

When	Before the extraction of each feature.
Uses	Displaying progress.

4.6.4.6. FeatureStop

When	After the extraction of each feature.
Uses	Displaying progress.

4.6.4.7. ClassifyStart

When	Before classification is started.
Uses	Hook in for custom machine learning tool.

4.6.4.8. ClassifyStop

When	After classification is done.
Uses	Displaying results in various forms.

4.6.4.9. TrainStart

When	Before training a corpus.
Uses	Hook machine learning tool into training event.

4.6.4.10. TrainStop

When	After training a corpus
Uses	Perform some action after training is done.

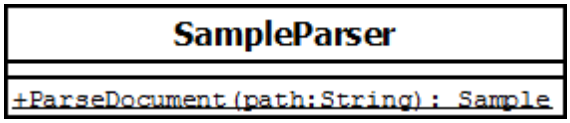
4.6.5. PlugIn Categories

Category	**Description**
Machine Learning	Tools used to classify and make a prediction about the author.
Feature Extraction	Tools used to extract individual features from a sample.
Visualization	Tools used to visualize various data in Stylo

4.7. Document Parsing

4.7.1. Sample Parser

The SampleParser class is responsible for parsing an input document into a Sample object.

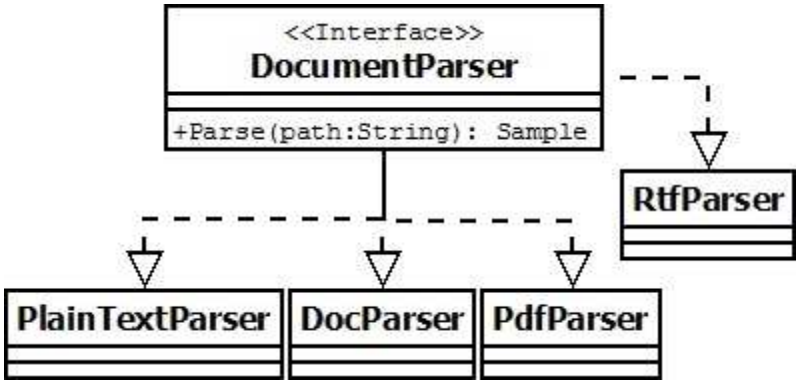


Methods

ParseDocument	**Input:** Path to document to parse **Output:** Sample object representing the specified document

4.7.2. DocumentParser Interface

The DocumentParser interface is a common interface between all document parsing classes.



Methods

Parse	**Input:** Path to document to parse **Output:** Sample object representing the specified document

4.7.2.1. PlainTextParser

The PlainText class is responsible for parsing a plain text document.

4.7.2.2. DocParser

The DocParser class is responsible for parsing a DOC document. OleFileIO_PL is used to extract the text from the DOC file.

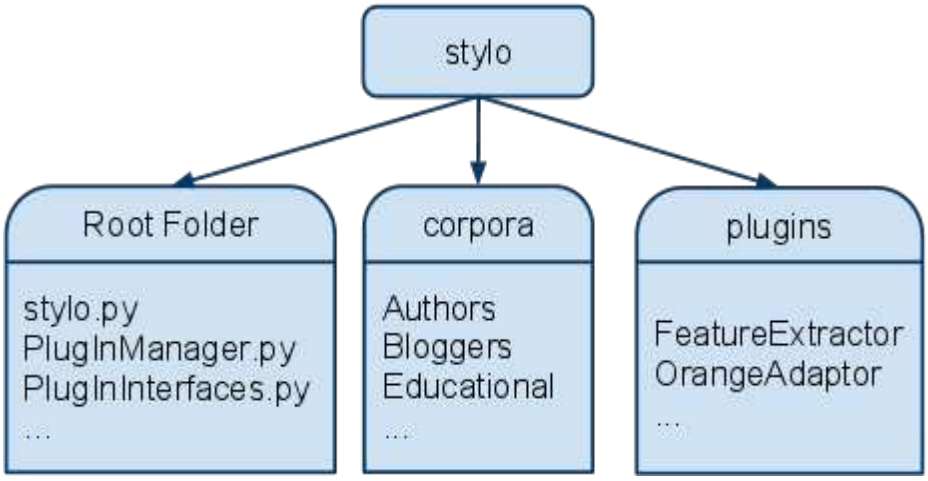
4.7.2.3. PdfParser

The PdfParser class is responsible for parsing a PDF document. pyPdf is used to extract the text from the PDF file.

4.7.2.4. RtfParser

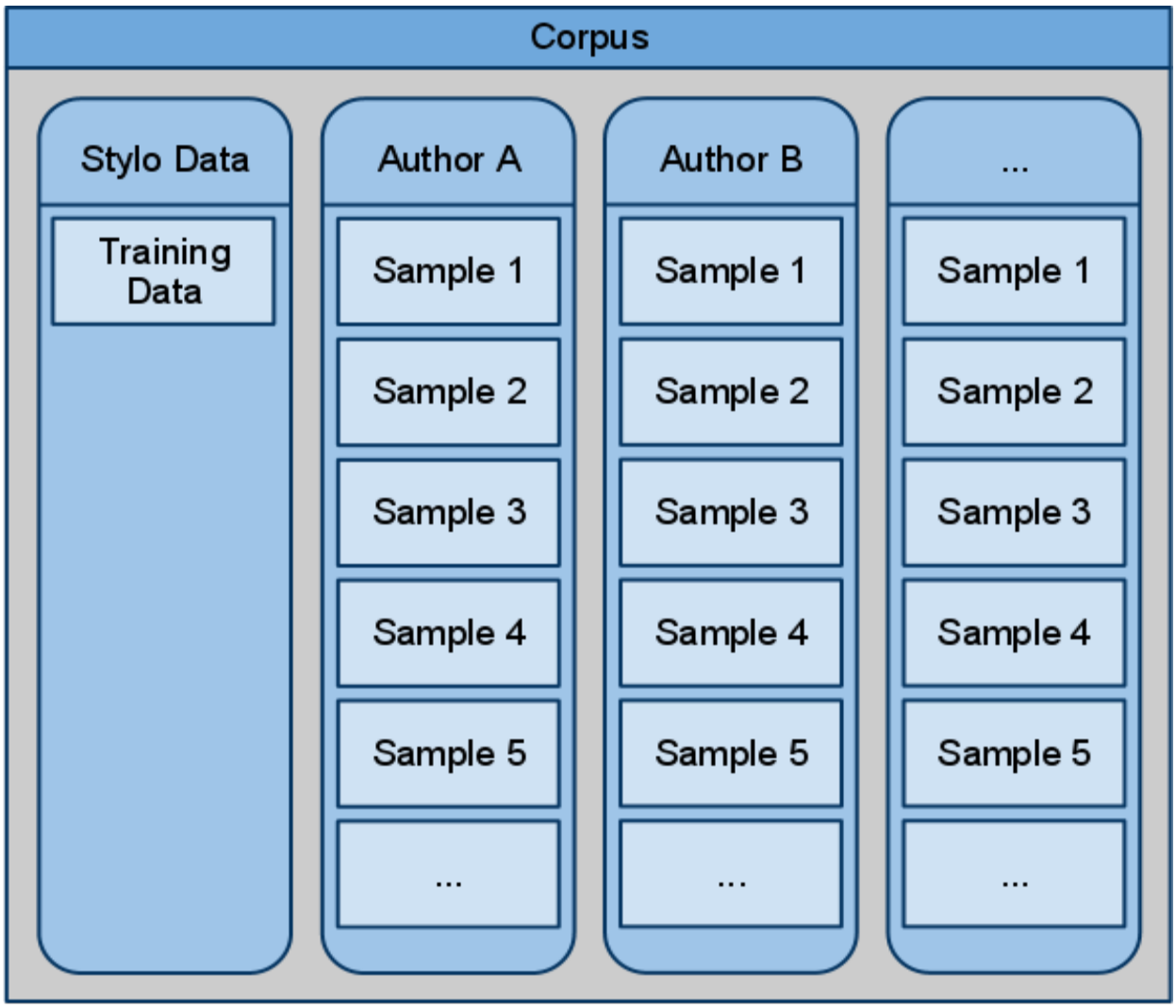
The RtfParser class is responsible for parsing an RTF document. pyRTF is used to extract the text from the RTF file.

4.8. File System Layout



4.8.1. Corpora Folder

On disk, a corpus is represented as a folder with many sub folders (in the case of an encrypted corpus, its a single archive). Each corpus' folder contains two types of folders, stylo folder and author folders. The name of the folder corresponds to the name of the corpus and must be unique. The corpora folder contains many individual corpora that are defined by the user.



4.8.1.1. Stylo Folder

The stylo folder resides inside the corpus' folder. This folder is different from the author folders as it contains information generated by Stylo that corresponds to the corpus, such as training data.

4.8.1.2. Author Folder

The author folder resides inside the corpus' folder to which it belongs. The name of the folder corresponds to alias Stylo uses to refer to the author whose documents are inside. This name need not be the real name of the author. This folder contains any number of samples of the author's writing in any file format that Stylo supports.

4.8.1.3. Corpus Encryption

If corpus encryption is enabled, the corpus and all data stored within will be contained inside of an encrypted zip file. The encryption key will be a user-supplied password for the corpus. This encryption will use the PyCrypto library.

4.8.1.4. Default Corpus

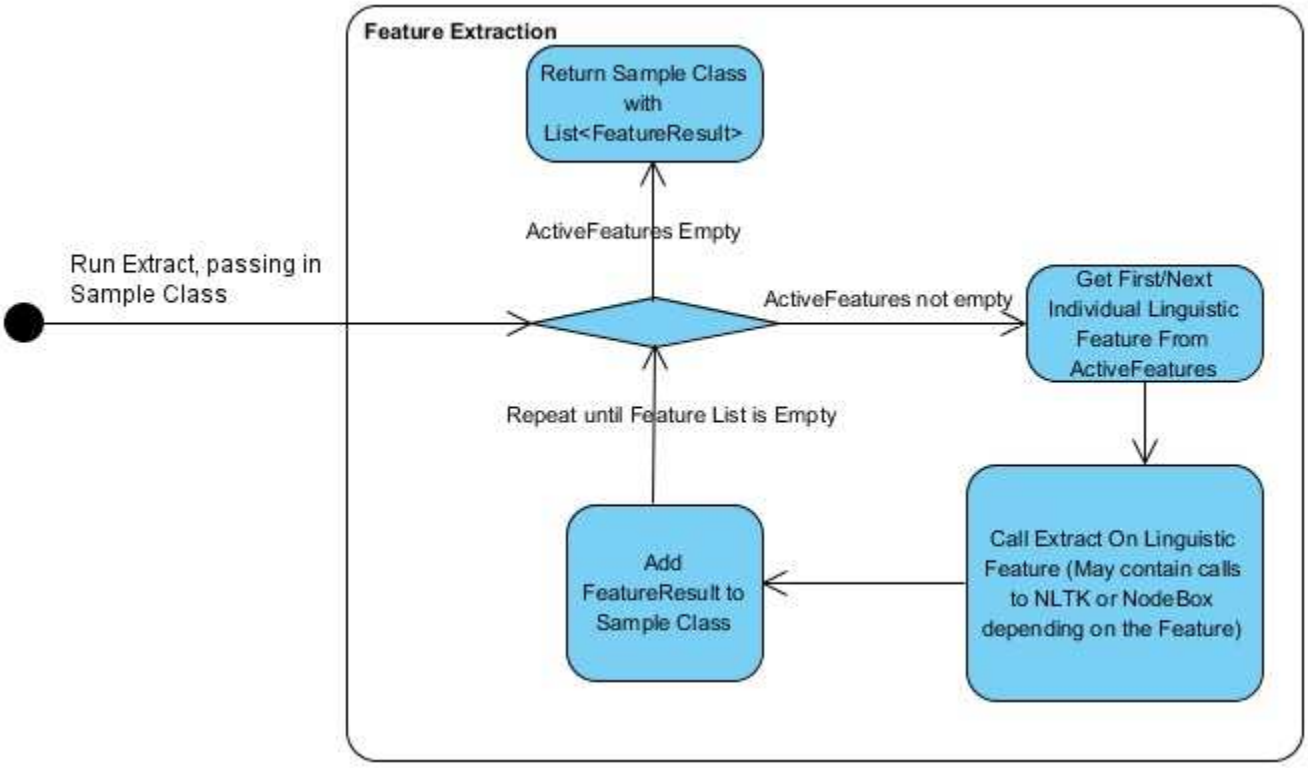
The software comes with a default corpus to use as an example for training and classification.

4.8.2. PlugIns Folder

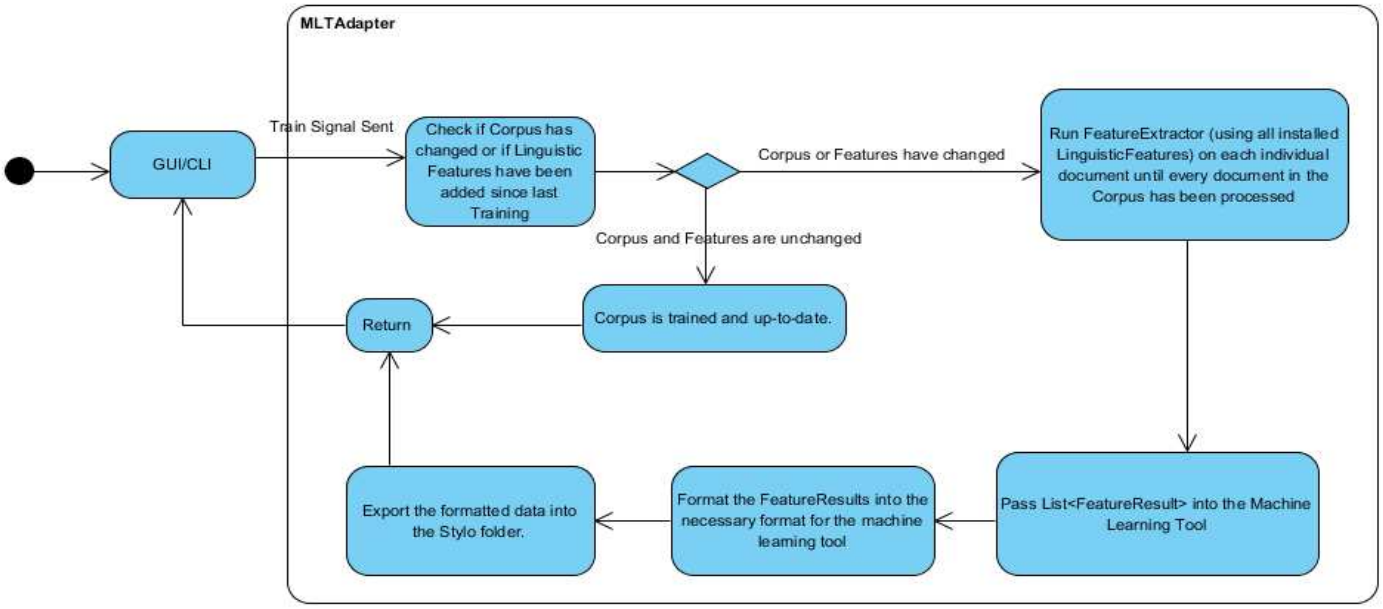
Each plugin is contained in a Python package which sits in the plugins folder. Plugins must follow the Python package specification.

4.9. Execution Flow

4.9.1. Feature Extraction



4.9.2. Training


MLTAdapter
GUI/CLI
Train Signal Sent
Check if Corpus has changed or if Linguistic Features have been added since last Training
Corpus or Features have changed
Run FeatureExtractor (using all installed LinguisticFeatures) on each individual document until every document in the Corpus has been processed
Corpus and Features are unchanged
Corpus is trained and up-to-date.
Return
Pass List<FeatureResult> into the Machine Learning Tool
Format the FeatureResults into the necessary format for the machine learning tool
Export the formatted data into the Stylo folder.

Appendix

A1. Traceability Matrix

Requirement Number	Design Component
4.2.1	4.5.1, 4.1.1.1.7
4.2.2.1	4.5.1
4.2.2.2	4.5.1
4.2.3.1	4.9.2
4.2.3.2	4.9.2
4.2.3.3	4.1.1.1.8
4.2.4	4.1.3
4.3.1.1.1	4.1.1.1.2
4.3.1.1.2	4.1.1.1.5
4.3.1.1.3	4.1.1.1.4
4.3.1.1.4	4.1.1.1.7
4.3.1.1.5	4.1.1.1.6
4.3.1.1.6	4.1.1.1.1
4.3.1.1.7	4.1.1.1.9
4.3.1.2	4.1.1.1.4
4.3.2.1	4.7.2.1
4.3.2.2	4.7.2.2
4.3.2.3	4.7.2.3
4.3.2.4	4.7.2.4
4.3.2.5	4.7.2

4.3.3.1	4.1.1.1.5
4.4.1.1.1	4.1.2.2.1
4.4.1.1.2	4.1.2.2.2
4.4.1.2.1	4.1.2.3.1
4.4.1.2.2	4.1.2.3.2
4.4.1.2.2.1	4.1.2.3.3
4.4.1.3.1	4.1.2.4.1
4.4.1.3.2	4.1.2.4.2
4.4.2.1	4.1.2
4.4.2.2	4.1.2
4.5.1.1	4.4.4.1
4.5.1.2	4.4.4.21
4.5.1.3	4.4.4.3
4.5.1.4	4.4.4.8
4.5.1.5	4.4.4.10
4.5.1.6	4.4.4.6
4.5.1.7	4.4.4.11
4.5.1.8	4.4.4.4
4.5.1.9	4.4.4.2
4.5.1.10	4.4.4.16
4.5.1.11	4.4.4.12
4.5.1.12	4.4.4.17
4.5.1.13	4.4.4.18
4.5.1.14	4.4.4.13
4.5.1.15	4.4.4.15
4.5.1.16	4.4.4.14
4.5.2	4.1.2.3.2

4.5.3	4.8.2
4.6.1	4.5.1
4.6.1.1	4.5.1
4.6.2	4.4.6
4.6.3.1	4.1.1.1.6, 4.5.1
4.6.3.1.1	4.1.1.1.2, 4.1.1.1.3, 4.1.2.3.2
4.6.3.1.2	4.1.1.1.2, 4.1.2.3.2
4.6.3.1.3	4.8.1.1, 4.9.2
4.6.3.2	4.1.1.1.6, 4.5.1
4.6.3.2.1	4.9.2
4.6.3.3	4.1.1.1.1
4.6.3.3.1	4.8.1.4
4.6.3.3.2	4.1.1.1.1, 4.8.1
5.2	4.6
5.3.1	4.1.2
5.3.2	4.1.2
5.3.3	4.1.2
5.3.4	4.1.2
5.4.1	2.3
5.4.2	2.3
5.4.3	2.3
5.5.1	2.3
5.5.2	2.3, 4.7.4
5.6	2.3

A2. Word Groups

Word Group Name	Words Included
Insightful	perhaps, think, believe, know, concept, thought, knew, formulate, thinking, belief, conceptualize, knowledge, knowing, concepts, grasp, understand, comprehend
Certainty	always, never, certain, definite, every
Sources	source, sources, site, sites, paper, papers, report, reports, article, articles, reference, references
Time Frame	now, currently, present, past, future, presently, archaic, old, new, out-dated, before
Tentative	maybe, might, guess, suppose, can, sometimes, tentative, perhaps
First Person Pronoun	I, me, my, mine, I've, I'd, I'm, I'll, myself
Second Person Pronoun	you, your, you're, yours, thou, yourself, yourselves
Third Person Singular Pronoun	he, she, his, her, him, hers, he'll, she'll he'd, she'd, he's, she's, herself, himself
Third Person Plural Pronoun	they, their, theirs, they'll, they'd, they've, theirself, themselves
Object Pronoun	it, its, it's, itself, those, these, that, this