Posh Predictive Text

Release 0.1.5

David Plummer

Sep 16, 2022

CONTENTS:

PowerShell predictive text for popular command line tools.

Productive messaging on the mobile phone arrived with the introduction of predictive text. The reduction in time needed to compose and send a message increases efficiency and allows the user to return to other activities more quickly.

Over time, the complexity of command line tools has increased. For new users the number of parameters and values can be overwhelming and for existing users entering the same commands can be repetitive and frustrating when small typing errors are made. Posh Predictive Text addresses these problems by suggesting text completions for parameters and, in some cases, parameter values.

CHAPTER

ONE

QUICK START

Posh Predictive Text is available from PowerShell Gallery and is installed and activated using the following instructions. A more detailed set of instructions are available on the installation page.

1.1 Prerequisites

Posh predictive text requires PowerShell version 7.2 or greater and PSReadLine version 2.2.6 or greater.

1.2 Download and install PowerShell module

Install PoshPredictive text from the PowerShell Gallery. This will download the module to the local user account. You may be asked for permission if you have not already set PowerShell gallery as trusted source.

```
Install-Module -name PoshPredictiveText
```

1.3 Update PowerShell profiles

Add the following commands to the PowerShell profile. To locate the PowerShell profile open a command prompt and type *$PROFILE*.

```
Set-PredictiveTextOption -RemoveCondaTabExpansion
Install-PredictiveText
```

1.3.1 Installation and configuration

Installing Posh Predictive Text

Install latest stable version into your python environment using *Install-Module*.

1.3.2 PowerShell Cmdlets

Posh Predictive Text uses cmdlets to implement features. These

Install-PredictiveText

SYNOPSIS

Installs Posh Predictive Text and activates the plugin and tab-expansion of command arguments.

SYNTAX

```
Install-PredictiveText [<CommonParameters>]
```

DESCRIPTION

Install Posh Predictive Text and activates both the PSReadLine plugin and enables tab-expansion of command line arguments.

A list of supported commands is provided by `Get-PredictiveTextOption -ListCommands`.

PARAMETERS

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

INPUTS

None

OUTPUTS

System.String

System.Management.Automation.CompletionResult

NOTES

RELATED LINKS

GitHub PoshPredictiveText Documentation

Get-PredictiveTextOption

SYNOPSIS

Gets options for Posh Predictive Text.

SYNTAX

Version

```
Get-PredictiveTextOption [-Version] [<CommonParameters>]
```

ListCommands

```
Get-PredictiveTextOption [-ListCommands] [<CommonParameters>]
```

PrintScript

```
Get-PredictiveTextOption [-PrintScript] [<CommonParameters>]
```

DESCRIPTION

Gets information about and options set for Posh Predictive Text argument completer.

PARAMETERS

-ListCommands

List commands supported with predictive text.

```
Type: SwitchParameter
Parameter Sets: ListCommands
Aliases: List, l
Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-PrintScript

Print PowerShell script used to initialise Powershell Predictive Text.

```
Type: SwitchParameter
Parameter Sets: PrintScript
Aliases: Print, p
Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Version

Returns the version.

```
Type: SwitchParameter
Parameter Sets: Version
Aliases: Ver, v
Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

INPUTS

None

OUTPUTS

System.String

System.Management.Automation.CompletionResult

NOTES

RELATED LINKS

GitHub PoshPredictiveText Documentation

Set-PredictiveTextOption

SYNOPSIS

Sets options for Posh Predictive Text.

SYNTAX

Logging

```
Set-PredictiveTextOption -LogFile <String> [-LogLevel <String>] [<CommonParameters>]
```

RemoveCondaTabExpansion

```
Set-PredictiveTextOption [-RemoveCondaTabExpansion] [<CommonParameters>]
```

DESCRIPTION

Sets options for Posh Predictive Text.

EXAMPLES

Example 1

```
Set-PredictiveTextOption -RemoveCondaTabExpansion
```

Conda installs its own tab-expansion within PowerShell. The conda code has higher precedence and prevents Posh Predictive Text providing completions for conda commands. This command removes the conda code and allows Posh Predictive Text to provide completions for conda.

Example 2

```
Set-PredictiveTextOption -LogFile 'C:\logiles\logfile.txt' -LogLevel INFO
```

Enables logging of diagnostic messages to a file. Messages are recorded against three levels: ERROR, WARN, INFO.

PARAMETERS

-LogFile

Enable logging to log file.

```
Type: String
Parameter Sets: Logging
Aliases:

Required: True
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-LogLevel

Level of information to log (INFO, WARN, ERROR). Default: ERROR.

```
Type: String
Parameter Sets: Logging
Aliases:
Accepted values: INFO, WARN, ERROR

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-RemoveCondaTabExpansion

Remove conda installed tab expansion.

```
Type: SwitchParameter
Parameter Sets: RemoveCondaTabExpansion
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

INPUTS

None

OUTPUTS

System.String

System.Management.Automation.CompletionResult

NOTES

RELATED LINKS

GitHub PoshPredictiveText Documentation

1.3.3 Development

PoshPredictiveText is developed using Visual Studio. The majority of the code is C# with a small amount of PowerShell scripting to register the native argument completer and remove conda tab-expansion code, if it is installed.

Coding Standards

Coding should follow Microsoft conventions <https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>.

Additional guidance provided by Google is used where appropriate <https://google.github.io/styleguide/csharp-style.html>.

The following standards are used in the code which may conflict with the above:

- *var* should be used sparingly and only when the type is **very** obvious. Use of *var* can make code difficult to review if the type is not clear.
- Indentation is four spaces (no tabs).
- Don't use underscore to indicate private, protected, internal and protected internal fields. It adds noise.
- *const* are indicated in capital case. Though this should change to follow Google guidelines.
- Opening braces do start on a new line.
- Do not use braces for single statements where it improves readability, e.g. if statements follwed by return to exit a method early *if (x is null) return null;*.

Testing

Testing uses Xunit, which should install from NuGet. Test coverage uses Fine Code Coverage, which is installed as an extension in Visual Studio.

Code should be covered with reasonable unit tests to validate that it works on multiple platforms.

All tests must pass before code is accepted.

PowerShell Help Documents

PowerShell has traditionally used MAML XML files for console help. These are difficult to prepare by hand. As an alternative, help files can be written in Markdown and then exported in required formats using the PowerShell PlatyPS module.

The PlatyPS module is described on GitHub <https://github.com/PowerShell/platyps>

The PlatyPS module must be downloaded and installed on the local development machine.

```
Install-Module -Name platyPS -Scope CurrentUser
Import-Module platyPS
```

The base documentation is already created within the folder *PowerShellHelpDocs*.

To create the external helpfiles run the following command whilist in the *PowerShellHelpDocs* folder.

```
New-ExternalHelp .\ -OutputPath en-US\
```

Source Code Documentation

Documentation is compiled using Sphinx to create a static website that is shared using Read The Docs. Source documents are generated using several tools and then collated within the sphinx documentation.

1. Source code documentation is generated by Doxygen and compiled into an XML file. This file is then parsed by Breathe/Sphinx-csharp and added to sphinx documents. All source code should be fully documented.
2. The PowerShell Help files are copied into the sphinx source directory and compiled into the sphinx documents.
3. The CHANGELOG.md is copied into the sphinx source directory and added to the sphinx documents.

- Doxygen: https://doxygen.nl/index.html
- Sphinx: https://www.sphinx-doc.org/en/master/
- Breathe: https://breathe.readthedocs.io/en/latest/index.html
- Sphinx-csharp: https://github.com/rogerbarton/sphinx-csharp
- Myst-parser: https://myst-parser.readthedocs.io/en/latest/
- Sphinx RTD Theme: https://sphinx-rtd-theme.readthedocs.io/en/stable/

1. Download and install Doxygen (https://doxygen.nl/download.html). If installing on Windows, add the doxygen bin folder to the path *C:Program Filesdoxygenbin*.
2. Create a Python environment for the Sphinx tools and install packages.

```
conda create -n poshpredictivetext python=3.10 sphinx sphinx_rtd_theme breathe myst-
↪parser
```

3. Install Sphinx-csharp to add c# support within Sphinx/Breathe.

```
pip install git+https://github.com/rogerbarton/sphinx-csharp.git
```

All documentation is generated during packaging and assumes that a 'conda' environment named *poshpredictivetext* is configured with the required dependencies.

API Documentation

Class: CommandAstVisitor

Note: Neither Doxygen nor sphinx-csharp support Python record types. Introduced in C# 9.

Once record types are recognised then details for Token will appear below.

The token is a record with the following values:

- readonly string Value - Text representing a token on the command line.
- readonly Type Type - The type of the token.

And the following read only properties which return true if the record is the indicated type:

- bool IsCommand
- bool IsCommandExpression
- bool IsCommandParameter
- bool IsStringConstantExpression

Warning: doxygenclass: Cannot find class "PoshPredictiveText::Token" in doxygen xml output for project "PoshPredictiveText" from directory: /home/docs/checkouts/readthedocs.org/user_builds/posh-predictive-text/checkouts/stable/docs/source/xml

`class PoshPredictiveText.CommandAstVisitor` : AstVisitor

Implements a visitor for the PowerShell abstract syntax tree. Reads the input no the command line and creates an ordered list of tokens representing user input.

Public Functions

`override` AstVisitAction `DefaultVisit` (Ast *ast*)

Default action if the token has not been processed by a more specific token handler. Adds the token text with a generic text type.

Param ast
Node in the abstract syntax tree.

Return
Flag to continue processing the abstract syntax tree.

`override` AstVisitAction `VisitCommand` (CommandAst *commandAst*)

Process commandAst node in the abstract syntax tree. Does not process this node and skips to next node in the tree.

Param commandAst
Node in the abstract syntax tree.

Return
Continue to next node.

`override` AstVisitAction `VisitCommandExpression` (CommandExpressionAst *commandExpressionAst*)

Process CommandExpressionAst node in the abstract syntax tree. Add a token to the list of tokens on the command line.

Param commandExpressionAst
Return
Continue to next node

`override` AstVisitAction `VisitCommandParameter` (CommandParameterAst *commandParameterAst*)

Process CommandParameterAst node in the abstract syntax tree. Add token to the list of tokens on the command line.

Param commandParameterAst
Node in the abstract syntax tree.

Return
Continue to next node.

`override` AstVisitAction `VisitConstantExpression` (ConstantExpressionAst *constantExpressionAst*)

Process ConstantExpressionAst node in the abstract syntax tree. Add token to the list of tokens on the command line.

Param constantExpressionAst
Node in the abstract syntax tree.

Return
Continue to next node.

`override` AstVisitAction `VisitStringConstantExpression` (StringConstantExpressionAst *stringConstantExpressionAst*)

Process stringConstantExpressionAst node in the abstract syntax tree.

Identifies Gnu/Posix formatted parameters which start with a double dash and reclassifies them as CommandParameterAst type.

Adds token to the list of tokens on the command line.

Param stringConstantExpressionAst
Node in the abstract syntax tree.

Return
Continue to next node.

Predictor: Predictor

`class PoshPredictiveText.Init` : IModuleAssemblyInitializer, IModuleAssemblyCleanup

Register the predictor on module loading and unregister it on module un-loading.

Public Functions

void **OnImport** ()

Gets called when assembly is loaded.

void **OnRemove** (PSModuleInfo *psModuleInfo*)

Gets called when the binary module is unloaded.

class PoshPredictiveText.Predictor : ICommandPredictor

PSReadLine plugin providing predictive text capabilities.

Public Functions

SuggestionPackage **GetSuggestion** (PredictionClient *client*, PredictionContext *context*, CancellationToken *cancellationToken*)

Get the predictive suggestions. It indicates the start of a suggestion rendering session.

Param client
Represents the client that initiates the call.

Param context
The PredictionContext object to be used for prediction.

Param cancellationToken
The cancellation token to cancel the prediction.

Return
An instance of SuggestionPackage.

bool **CanAcceptFeedback** (PredictionClient *client*, PredictorFeedbackKind *feedback*)

Gets a value indicating whether the predictor accepts a specific kind of feedback.

Param client
Represents the client that initiates the call.

Param feedback
A specific type of feedback.

Return
True or false, to indicate whether the specific feedback is accepted.

void **OnSuggestionDisplayed** (PredictionClient *client*, uint *session*, int *countOrIndex*)

One or more suggestions provided by the predictor were displayed to the user.

Param client
Represents the client that initiates the call.

Param session
The mini-session where the displayed suggestions came from.

Param countOrIndex
When the value is greater than 0, it's the number of displayed suggestions from the list returned in *session* , starting from the index 0. When the value is less than or equal to 0, it means a single suggestion from the list got displayed, and the index is the absolute value.

void **OnSuggestionAccepted** (PredictionClient *client*, uint *session*, string *acceptedSuggestion*)

The suggestion provided by the predictor was accepted.

Param client
Represents the client that initiates the call.

Param session
Represents the mini-session where the accepted suggestion came from.

Param acceptedSuggestion
The accepted suggestion text.

void **OnCommandLineAccepted** (PredictionClient *client*, IReadOnlyList<string> *history*)

A command line was accepted to execute. The predictor can start processing early as needed with the latest history.

Param client
Represents the client that initiates the call.

Param history
History command lines provided as references for prediction.

void **OnCommandLineExecuted** (PredictionClient *client*, string *commandLine*, bool *success*)

A command line was done execution.

Param client
Represents the client that initiates the call.

Param commandLine
The last accepted command line.

Param success
Shows whether the execution was successful.

Properties

Guid **Id** { **get; set;** }

Gets the unique identifier for a subsystem implementation.

string? **Name** { **get; set;** }

Gets the name of a command for which suggestion are being made.

string **Description** { **get; set;** }

Gets the description of a subsystem implementation.

Cmdlet: GetPredictiveText

class PoshPredictiveText.GetPredictiveText : PSCmdlet

The Posh Predictive Text cmdlet provides suggested completions for command arguments in PowerShell.

Properties

string? **WordToComplete** { **get; set;** }

Gets or sets the partial word provided before the user pressed Tab.

CommandAst? **CommandAst** { **get; set;** }

Gets or sets the Abstract Syntax Tree (AST) for the current input line.

int? **CursorPosition** { **get; set;** }

Gets or sets the position of the cursor when tab was pressed.

Cmdlet: GetPredictiveTextOption

class PoshPredictiveText.GetPredictiveTextOption : PSCmdlet

The Posh Predictive Text cmdlet provides suggested completions for command arguments in PowerShell.

Properties

SwitchParameter **Version** { **get; set;** }

Gets or sets the parameter requesting the version to be display.

SwitchParameter **ListCommands** { **get; set;** }

Gets or sets the switch indicating List, ListCommand or L flag supported commands.

SwitchParameter **PrintScript** { **get; set;** }

Gets or sets a flag to print the PowerShell script used to initialise tab-exapansion of arguments.

Cmdlet: InstallPredictiveText

class PoshPredictiveText.InstallPredictiveText : PSCmdlet

The Posh Predictive Text cmdlet provides suggested completions for command arguments in PowerShell.

Cmdlet: SetPredictiveTextOption

class PoshPredictiveText.SetPredictiveTextOption : PSCmdlet

The Posh Predictive Text cmdlet provides suggested completions for command arguments in PowerShell.

Properties

string? **LogFile** { **get; set;** }

Gets or sets the file to which messages will be written.

string? **LogLevel** { **get; set;** }

Gets or sets the flag indicating *Init* command generates and invokes a PowerShell script to add the cmdlet as an Argument Resolver in PowerShell.

SwitchParameter **RemoveCondaTabExpansion** { **get; set;** }

Gets or sets the flag that conda tabexpansion should be removed.

1.3.4 Upgrading and change log

Recent versions of Posh Predictive Text have a corresponding git tag for each version released to PowerShell Gallery.

Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

[0.1.1] - 2022-09-16

Changed

- Improved documentation for PowerShell Gallery.

[0.1.0] - 2022-09-16

Added

- Initial release.
- New text completions: `conda`.

Changed

- Nothing