NormAPI: An API for Normalizing Filipino Shortcut Texts
User's Manual

A Thesis
Presented to
the Faculty of the College of Computer Studies
De La Salle University

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science in Computer Science

by

Cuevas, Justin Gems G.
Magat, Enrico Darwin S.
Nocon, Nicco Louis S.
Suministrado, Peter Gabriel D.

Ms. Charibeth Cheng
Faculty Adviser

September 2, 2014

Table of Contents

1 Introduction

NormAPI is an API or Application Programming Interface which functions mainly to normalize or transform Filipino shortcut texts back to its original or correct form through the use of statistical machine translation and/or dictionary substitution approaches.

1.1 System Requirements

This subsection lists the minimum hardware and software requirements needed to properly execute the system.

Windows	
Operating System	Windows 7 (32-bit or 64-bit)
CPU	Intel ® Core™2 Quad
[*]Memory	2GB
Hard Disk Drive	9GB

1.2 Conventions

This subsection describes the words or phrases in the manual that are given emphasis or distinction in all sections of the manual.

Here is the list of conventions that was made in the document:

- Default font: Arial; Default size: 11;
- Function names have a font style of `Courier New` and are in **boldface**.
- Words that have a font size of 10 can either be of the following:
  - Footnotes
  - Table Entries
- Words that have been *italicized* or are in *italic* is either of the following:
  - Name of a folder
  - Name of a file
  - A file path or destination path
  - Declaration part of the parameters column in the NormAPI Methods Table
  - Introductory messages in the sections and subsections
- Words that have single quotation marks ('<word>') pertain to a button.
- Scripts/commands have a font style of `Lucida Console` and a font size of 10.
- Words that have a font style of Courier New and a font size of 10 are examples of code on how NormAPI works in the Sample Code subsection.
- Words that are in **boldface** are either of the following:
  - Title of the sections and subsections
  - Column names
  - Table or Figure number

[*]In 100,000 sentence entry corpus, 4GB of memory is advised.

1.3 Installation

This subsection contains instructions on how to install the system, and the list of necessary files and their respective directories.

Needed for installation:

Cygwin Installation Files folder – Moses needs to be run on a Linux environment. For Windows users, Cygwin allows the users to use Moses in a Windows platform since it provides a Linux-like environment.

Giza++ Installation Files folder – Giza++ is a requirement of Moses for training.

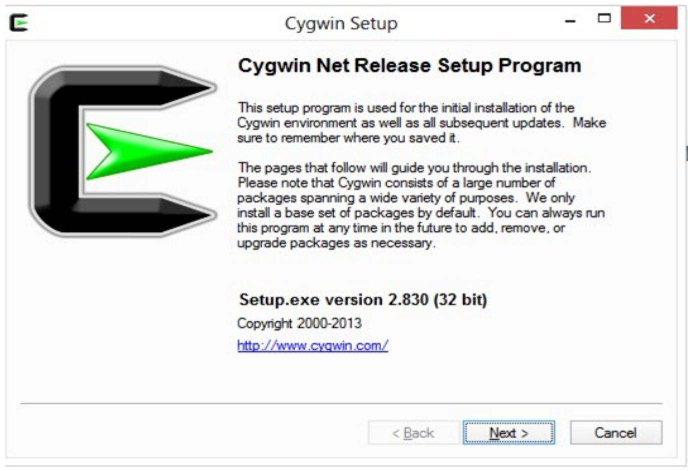
Boost Installation Files folder – Boost is a requirement of Moses for training.

SRILM Installation Files folder – SRILM is a toolkit for building and applying statistical language models (LMs), primarily for use in speech recognition, statistical tagging and segmentation, and machine translation. LM is used in training to generate the phrase model.

Moses Installation Files folder – Moses is a statistical machine translation system that allows you to automatically train phrase models for any language pair. Moses also provides the phrase table that is used in translation.

1.3.1 [*]Cygwin:

1. Open the *Cygwin Installation Files* folder.
2. Look for the *setup-x86.exe* inside the folder and double click it.
3. After double clicking the file, this window will appear. Click 'Next'.



Cygwin Net Release Setup Program

This setup program is used for the initial installation of the Cygwin environment as well as all subsequent updates. Make sure to remember where you saved it.

The pages that follow will guide you through the installation. Please note that Cygwin consists of a large number of packages spanning a wide variety of purposes. We only install a base set of packages by default. You can always run this program at any time in the future to add, remove, or upgrade packages as necessary.

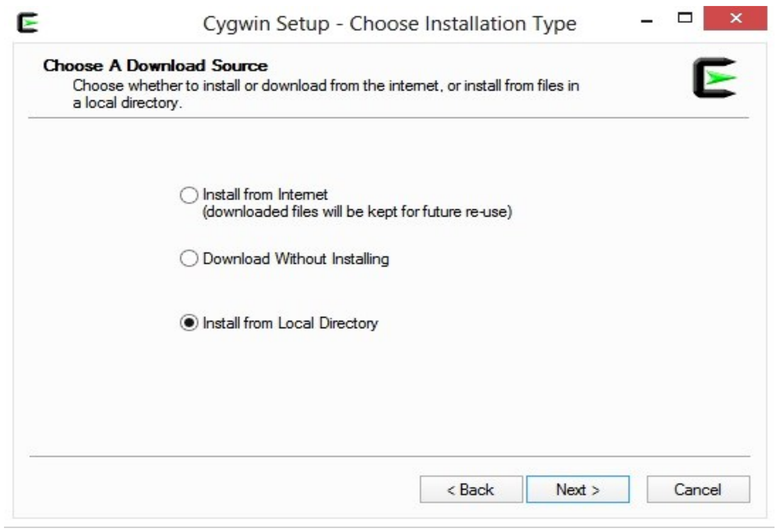
Setup.exe version 2.830 (32 bit)

Copyright 2000-2013

http://www.cygwin.com/

[*]If ever a warning about versions or dependencies of Cygwin is encountered, click 'OK'. If another version of Cygwin was previously installed in your computer, errors might occur during the installation. Before installing all of the components of Cygwin, instead of choosing Install in Step #7, choose Uninstall to solve this problem. Do this to all components of Cygwin.

4. [*]Choose the *Install from Local Directory* option. Click ‘Next’



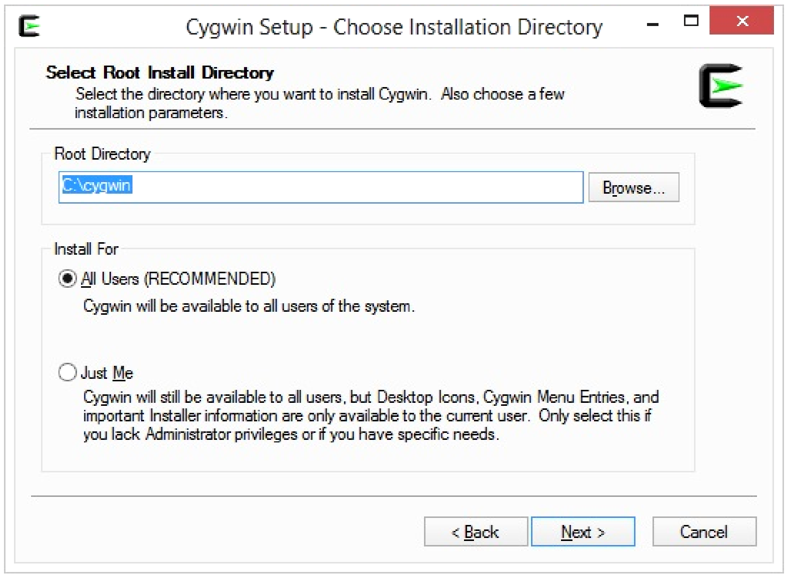
Cygwin Setup - Choose Installation Type

Choose A Download Source

Choose whether to install or download from the internet, or install from files in a local directory.

- ◯ Install from Internet (downloaded files will be kept for future re-use)
- ◯ Download Without Installing
- ● Install from Local Directory

5. Choose a directory where Cygwin would be installed. Click ‘Next’.



Cygwin Setup - Choose Installation Directory

Select Root Install Directory

Select the directory where you want to install Cygwin. Also choose a few installation parameters.

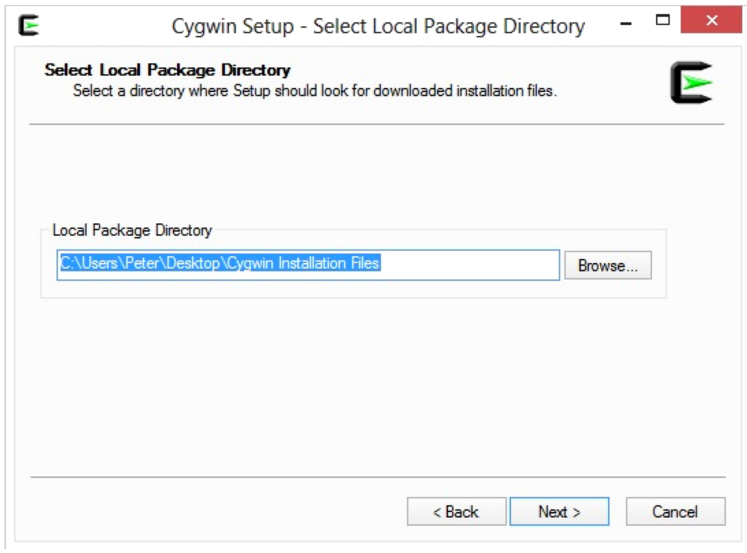
Root Directory: C:\cygwin

Install For

- ● All Users (RECOMMENDED)
  Cygwin will be available to all users of the system.
- ◯ Just Me
  Cygwin will still be available to all users, but Desktop Icons, Cygwin Menu Entries, and important Installer information are only available to the current user. Only select this if you lack Administrator privileges or if you have specific needs.

[*] Installation can also be done by choosing the first option, *Install from Internet*.

6. Browse where the *Cygwin Installation Files* folder is located then click 'Next'.



Cygwin Setup - Select Local Package Directory

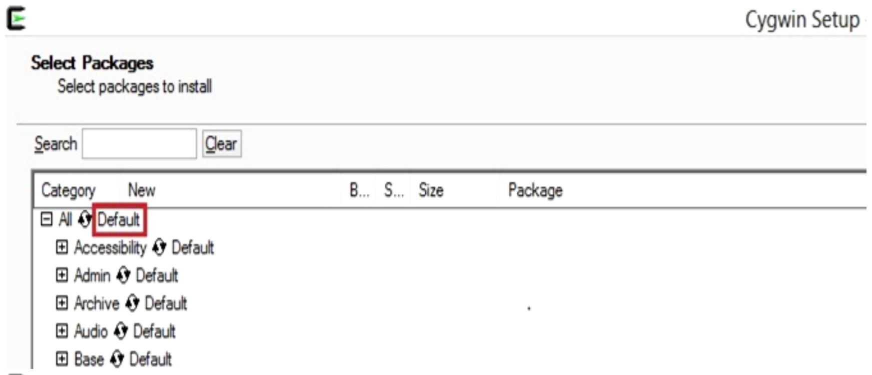
Select Local Package Directory
Select a directory where Setup should look for downloaded installation files.

Local Package Directory: C:\Users\Peter\Desktop\Cygwin Installation Files

Select a folder inside: *C:\<your-path>\Cygwin Installation Files*

e.g. C:\<your-path>\Cygwin Installation Files\Bzip

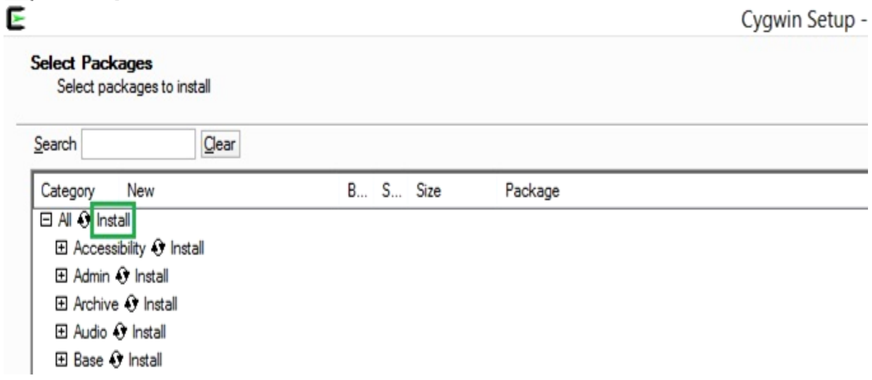
7. The installer will now ask you to choose which packages are to be installed, click 'Default' which is beside 'All' until 'Default' turns to 'Install'. After that click 'Next'.



Cygwin Setup

Select Packages
Select packages to install

```
⊟ All Default
  ⊞ Accessibility Default
  ⊞ Admin Default
  ⊞ Archive Default
  ⊞ Audio Default
  ⊞ Base Default
```

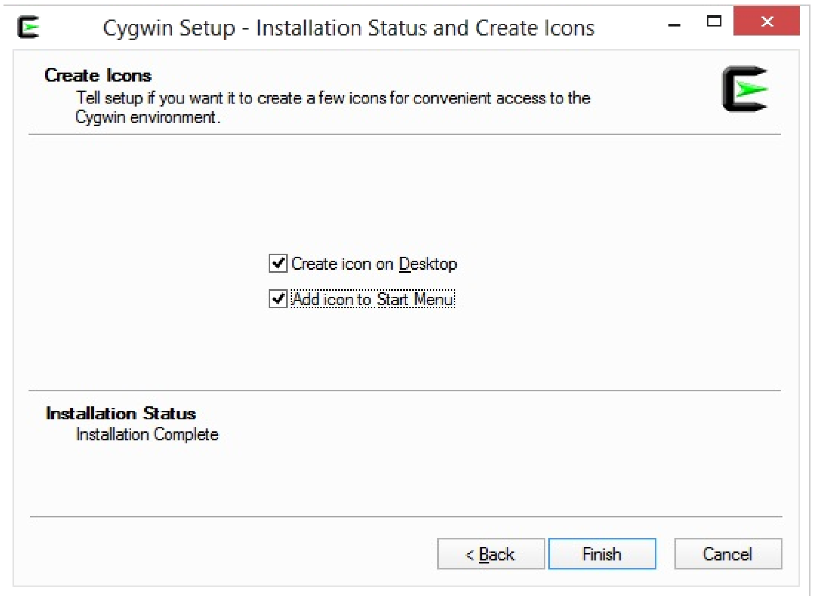


Cygwin Setup -

Select Packages
Select packages to install

```
⊟ All Install
  ⊞ Accessibility Install
  ⊞ Admin Install
  ⊞ Archive Install
  ⊞ Audio Install
  ⊞ Base Install
```

8. Wait for Cygwin to finish installing.
9. When the installation is done, click 'Finish'.



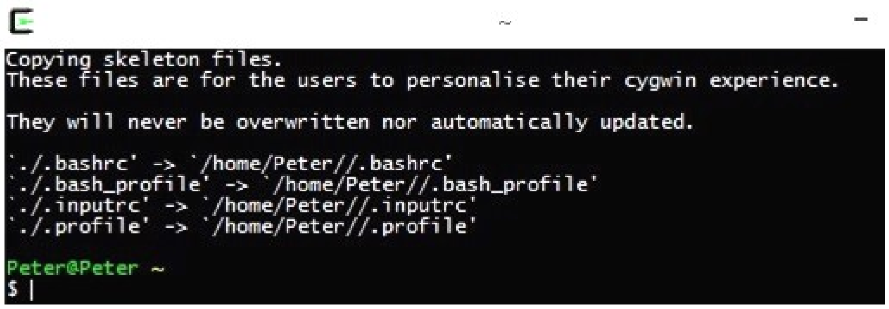
Create Icons
Tell setup if you want it to create a few icons for convenient access to the Cygwin environment.

[✓] Create icon on Desktop
[✓] Add icon to Start Menu

Installation Status
Installation Complete

10. Repeat the process and change the field in 'Step 6' into the location of one of the following folders: (all folders are located inside the *Cygwin Installation Files* folder)
    - Basics (preferable to be first in installation)
    - Boost
    - BZ
    - BZip
    - Gcc part 1, part 2, part 3
    - Libboost
    - Misc
    - Read
    - SRILM Requirements
    - Zlib part 1, part 2, part 3

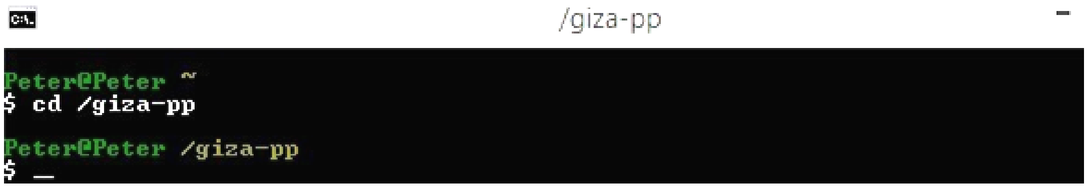
11. When all are folders are now installed to Cygwin, go to *C:\cygwin\* and double click *Cygwin.bat* to initialize user settings.



```
Copying skeleton files.
These files are for the users to personalise their cygwin experience.

They will never be overwritten nor automatically updated.

`./.bashrc' -> `/home/Peter//.bashrc'
`./.bash_profile' -> `/home/Peter//.bash_profile'
`./.inputrc' -> `/home/Peter//.inputrc'
`./.profile' -> `/home/Peter//.profile'

Peter@Peter ~
$ |
```

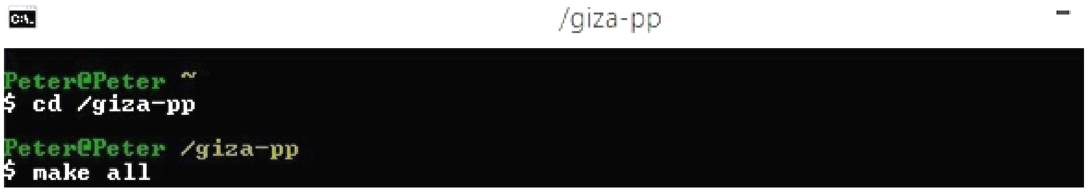
1.3.2 Giza++

1. Open the *Giza++ Installation Files* folder.
2. Extract *giza-pp-v1.0.7.tar.gz* to the current folder until the folder *giza-pp* appears.
3. Move *giza-pp* to *C:\cygwin\*
4. Open *Cygwin.bat* in *C:\cygwin\*
5. Enter: `cd /giza-pp` to change the directory



```
/giza-pp

Peter@Peter ~
$ cd /giza-pp

Peter@Peter /giza-pp
$ _
```

6. Enter: `make all`

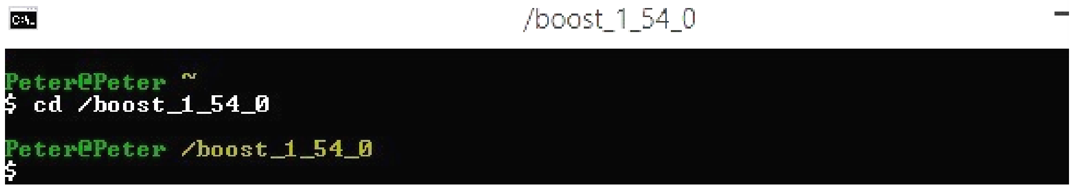


```
/giza-pp

Peter@Peter ~
$ cd /giza-pp

Peter@Peter /giza-pp
$ make all
```

7. Wait for the script to finish and after that Giza++ is now installed.

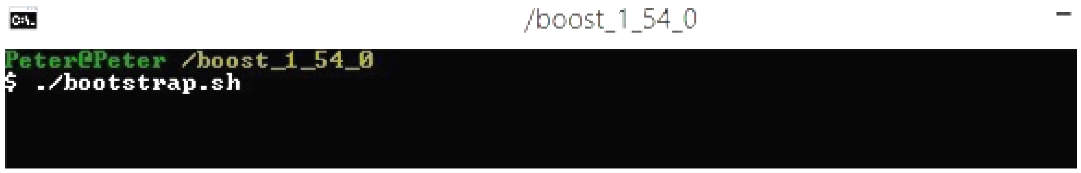
1.3.3 Boost

1. Open *Boost Installation Files* folder.
2. Extract *boost_1_54_0.7z* to the current folder until the folder *boost_1_54_0* appears.
3. Move *boost_1_54_0* to *C:\cygwin\*
4. Open *Cygwin.bat*
5. Enter: `cd /boost_1_54_0` to change directory



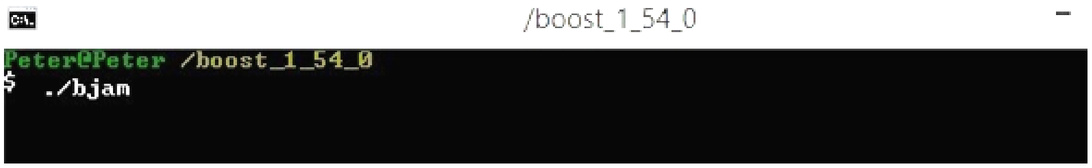
```
/boost_1_54_0

Peter@Peter ~
$ cd /boost_1_54_0

Peter@Peter /boost_1_54_0
$
```

6. Enter: `./bootstrap.sh`



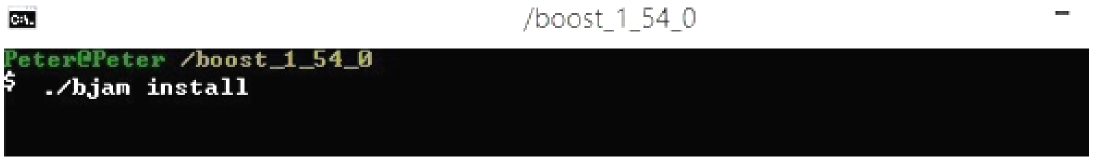
```
/boost_1_54_0

Peter@Peter /boost_1_54_0
$ ./bootstrap.sh
```

7. Enter: `./bjam`



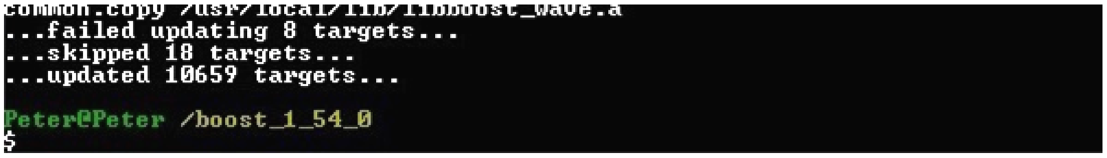
```
/boost_1_54_0
Peter@Peter /boost_1_54_0
$  ./bjam
```

8. Enter: `./bjam install`



```
/boost_1_54_0
Peter@Peter /boost_1_54_0
$  ./bjam install
```

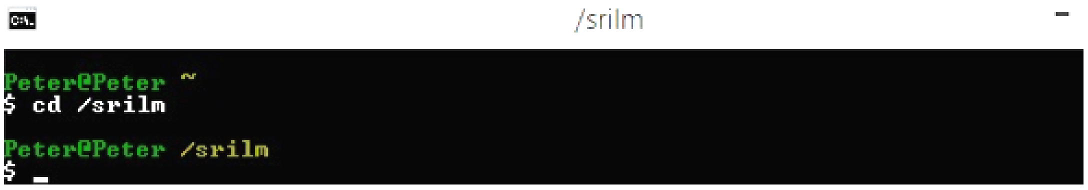
9. After the script, it should result to with 8 failed targets and 18 skipped targets



```
common.copy /usr/local/lib/libboost_wave.a
...failed updating 8 targets...
...skipped 18 targets...
...updated 10659 targets...

Peter@Peter /boost_1_54_0
$
```

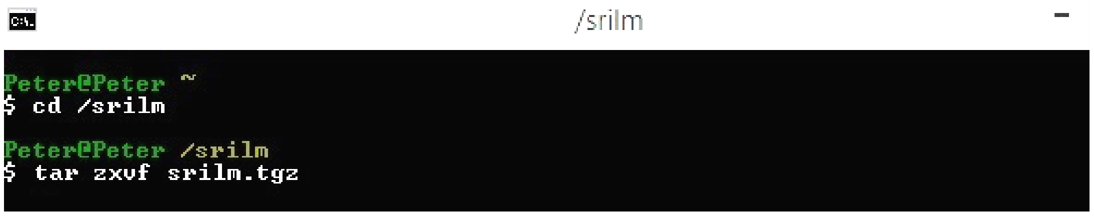
1.3.4 SRILM

1. Create directory or folder for SRILM in *C:\cygwin\* named *srilm*
2. Copy *srilm.tgz* from the *SRILM Installation Files* Folder then paste it to *C:\cygwin\srilm\*
3. Open *Cygwin.bat*
4. Enter: `cd /srilm`



```
/srilm
Peter@Peter ~
$ cd /srilm

Peter@Peter /srilm
$ _
```

5. Extract *srilm.tgz* through the script: `tar zxvf srilm.tgz`



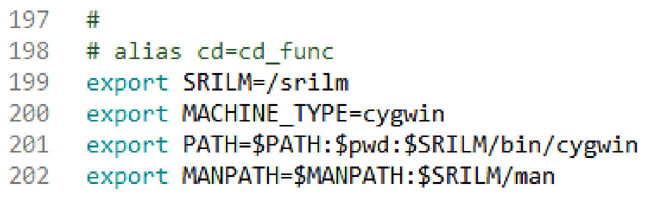
```
/srilm
Peter@Peter ~
$ cd /srilm

Peter@Peter /srilm
$ tar zxvf srilm.tgz
```

6. Close *Cygwin.bat*

7. Edit: *C:\cygwin\home\<your-name>\.bashrc*

   Add lines:

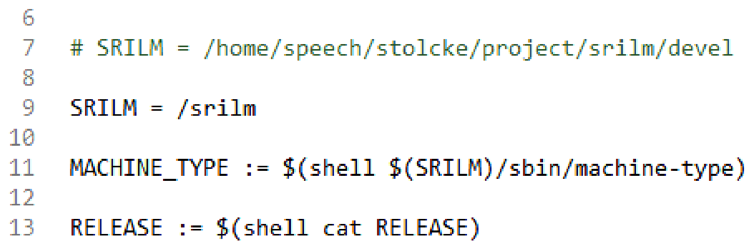
```
export SRILM=/srilm
export MACHINE_TYPE=cygwin
export PATH=$PATH:$pwd:$SRILM/bin/cygwin
export MANPATH=$MANPATH:$SRILM/man
```



```
#
# alias cd=cd_func
export SRILM=/srilm
export MACHINE_TYPE=cygwin
export PATH=$PATH:$pwd:$SRILM/bin/cygwin
export MANPATH=$MANPATH:$SRILM/man
```

8. Edit: C:\cygwin\srilm\Makefile

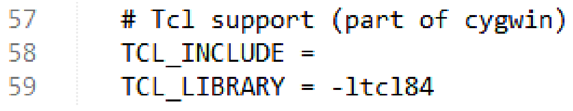
   Add line: `SRILM = /srilm`



```

# SRILM = /home/speech/stolcke/project/srilm/devel

SRILM = /srilm

MACHINE_TYPE := $(shell $(SRILM)/sbin/machine-type)

RELEASE := $(shell cat RELEASE)
```

9. Edit: C:\cygwin\srilm\common\Makefile.machine.cygwin

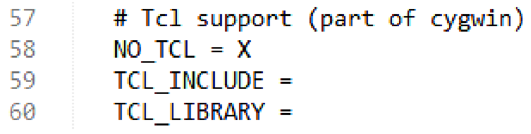
   Remove the group: `# Tcl support (part of cygwin)`



```
# Tcl support (part of cygwin)
TCL_INCLUDE =
TCL_LIBRARY = -ltcl84
```

   Replace with:

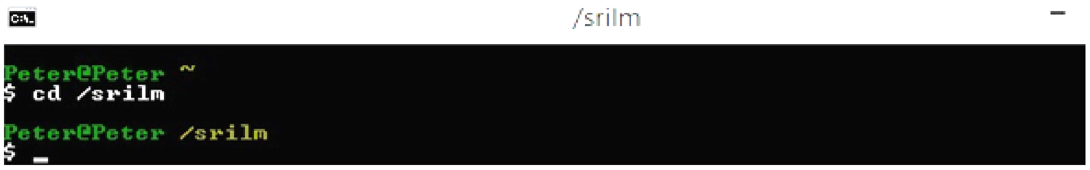
```
# Tcl support (part of cygwin)
NO_TCL = X
TCL_INCLUDE =
TCL_LIBRARY =
```



```
# Tcl support (part of cygwin)
NO_TCL = X
TCL_INCLUDE =
TCL_LIBRARY =
```

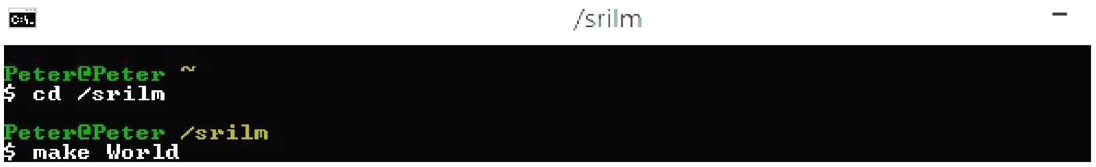
10. Open *Cygwin.bat*

11. Enter: `cd /srilm`



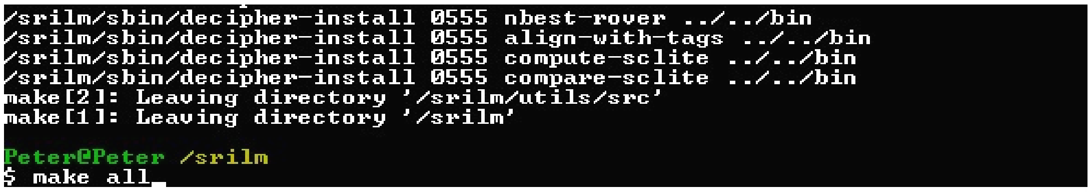
```
/srilm
Peter@Peter ~
$ cd /srilm

Peter@Peter /srilm
$ _
```

12. Enter: `make World`



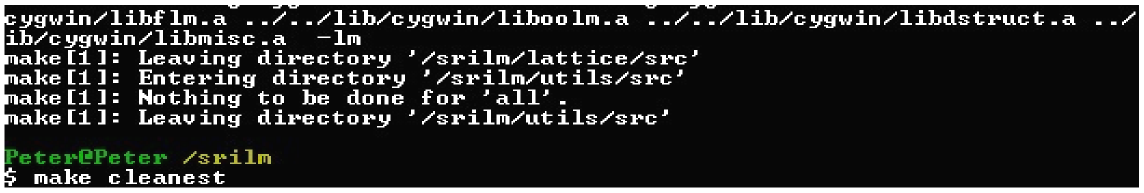
```
/srilm
Peter@Peter ~
$ cd /srilm

Peter@Peter /srilm
$ make World
```

13. Enter: `make all`



```
/srilm/sbin/decipher-install 0555 nbest-rover ../../bin
/srilm/sbin/decipher-install 0555 align-with-tags ../../bin
/srilm/sbin/decipher-install 0555 compute-sclite ../../bin
/srilm/sbin/decipher-install 0555 compare-sclite ../../bin
make[2]: Leaving directory '/srilm/utils/src'
make[1]: Leaving directory '/srilm'

Peter@Peter /srilm
$ make all_
```

14. Enter: `make cleanest`
15. After the script, it should close by leaving the directory



```
cygwin/libflm.a ../../lib/cygwin/liboolm.a ../../lib/cygwin/libdstruct.a ../
ib/cygwin/libmisc.a  -lm
make[1]: Leaving directory '/srilm/lattice/src'
make[1]: Entering directory '/srilm/utils/src'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/srilm/utils/src'

Peter@Peter /srilm
$ make cleanest
```

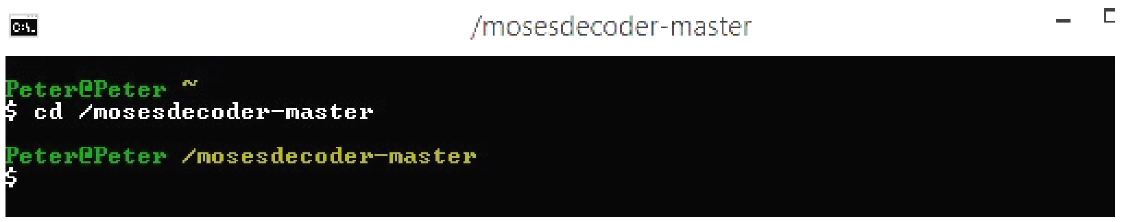
Reference of SRILM Installation: Introduction to SRILM Toolkit.pdf by Berlin Chen*

1.3.5 Moses

1. Open the *Moses Installation Files* folder.
2. Extract *mosesdecoder-master.zi*p to the current folder.
3. Move *mosesdecoder-master* folder to *C:\cygwin* (make sure that the mosesdecoder-master folder contains the files of moses and not another moses folder)
4. Open *Cygwin.bat*

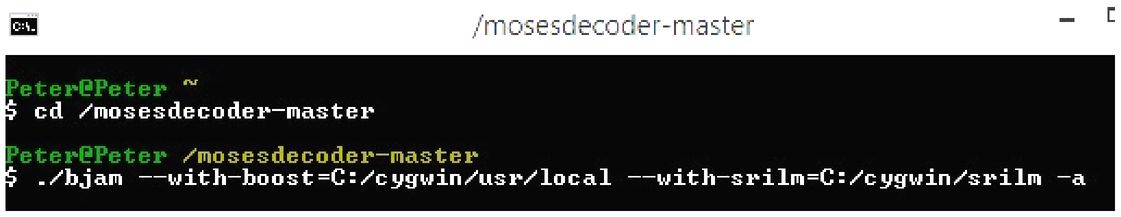
*http://www.cs.brandeis.edu/~cs114/CS114_docs/SRILM_Tutorial_20080512.pdf

5. Enter: `cd /mosesdecoder-master`



```
Peter@Peter ~
$ cd /mosesdecoder-master

Peter@Peter /mosesdecoder-master
$
```

6. Enter: `./bjam --with-boost=C:/cygwin/usr/local --with-srilm=C:/cygwin/srilm -a`



```
Peter@Peter ~
$ cd /mosesdecoder-master

Peter@Peter /mosesdecoder-master
$ ./bjam --with-boost=C:/cygwin/usr/local --with-srilm=C:/cygwin/srilm -a
```

7. After the script is processed, it should close with success

1.4 Other Required Actions

The following are required actions after installing all components:

- Create a folder with the name “tools” inside the *mosesdecoder-master* folder. The following are files that the tools folder should contain: *GIZA++.exe*, *mkcls.exe* and *snt2cooc.out*. Copy those files and paste them inside the tools fofder.=
  - Both *GIZA++.exe* and *snt2cooc.out* are located in this directory: *C:\cygwin\giza-pp\GIZA++-v2*
  - *mkcls.exe* is located in this directory: *C:\cygwin\giza-pp\mkcls-v2*
- Append these directories in the Path of the System Variables in the Environment Variables of your computer: *C:\cygwin\mosesdecoder-master\bin\;C:\cygwin\bin\;C:\cygwin\srilm\bin\cygwin\;*

1.5 [*]Possible Errors

Here are possible instances or scenarios where errors may occur during installation:

- During Cygwin installation, if a previous or old version of Cygwin was installed, it will most likely cause an error in the individual installation of its components.
- If Cygwin was not installed properly, all the remaining installation files would be impossible to install.
- If the extraction of Giza++ was not done or if the extracted folder was not moved into the Cygwin folder or the scripts/commands were not executed through *Cygwin.bat*.

[*] Possible errors while using NormAPI is found in Section 4.1

- If the extraction of Boost was not done or if the extracted folder was not moved into the Cygwin folder or the scripts/commands were not executed through *Cygwin.bat*.
- If a folder for srilm was not created in the Cygwin directory or execution of scripts were not done properly or the files mentioned were not changed.
- If the extraction of Moses was not done or if the extracted folder was not moved into the Cygwin folder or the scripts/commands were not executed through *Cygwin.bat*.
- Basically, if the instructions in installing all components were not followed properly, expect errors will occur during installation.

Here are possible instances or scenarios where errors may occur when using the installed components:

- If one of the installations was not done correctly, Moses will not function properly.
- If the NormAPI file was not included in the library or dependency of the program, the NormAPI functions will be unusable.
- If the NormAPI was not imported explicitly in the code of the program, the NormAPI functions will be unusable.

2 Getting Started

This section has subsections that discusses 2.1 the instructions in how another program can use NormAPI and 2.2 the details of TestingNormAPI.

2.1 Using NormAPI as an API

NormAPI is a Java Application Program Interface (API) used in normalizing shortcut text through four normalization variations. An API per se is a set of routines, protocols, and tools for building software applications. Since NormAPI is an API that means it will be used by developers in creating a software application.

For a developer to use API in his or her software, they must first have a copy *NormAPI.jar* and must include it as one of the libraries or dependencies (for Maven) in their software. They must also insert this piece of code in the class where the API will be needed: *import normapi.NormAPI;.* Normally, in some IDEs importing the .jar in a library automatically imports .jar on the code.

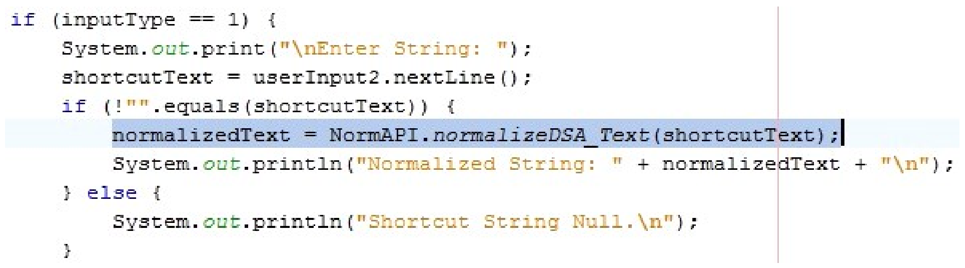
2.2 TestingNormAPI Functions

TestingNormAPI is a program with a console interface that accesses the functionalities of NormAPI such as normalizing using the four variations, setting and appending the word dictionary, training data, setting the configuration file and calculating the BLEU score. The normalization options provide the choice for users whether to directly input a sentence or load a corpus file. This program has only one class named TestingNormAPI which is located in the testingnormapi source package. This class has no super class and no specified properties. Its further details are shown in Table 1.

Table 1: TestingNormAPI Class Details

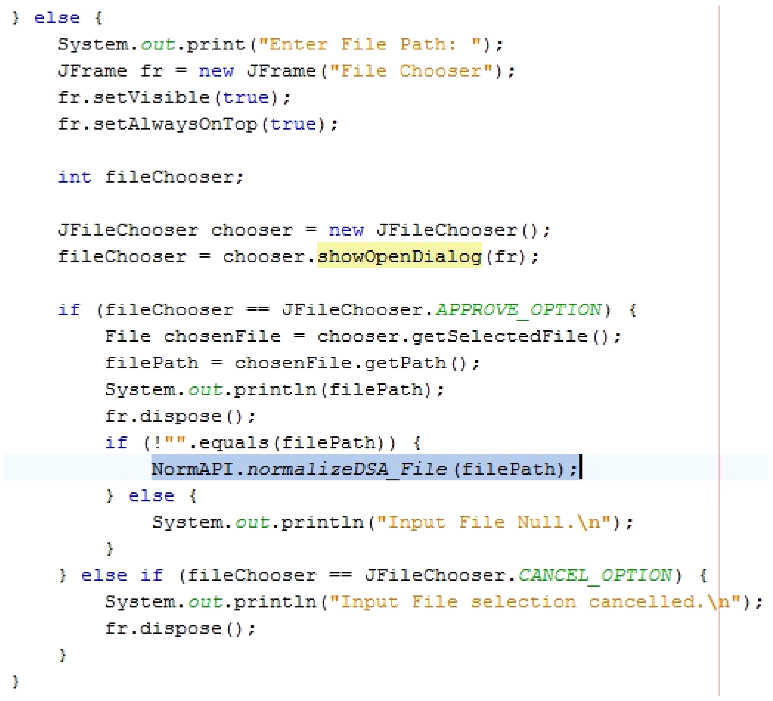
Method Name	Description	Parameters	Result Type	Constraint
`Main`	This method displays the console interface of the program and calls the rest of the methods in this class and methods in NormAPI such as **`NormAPI.normalizeDSA_Text`** and **`NormAPI.normalizeSMT_Text`**.	String[] args	Void	Public
`setDictionary`	This method changes the word dictionary accessed by DSA.	None	Void	Public
`setConfigurationFile`	This method changes the Moses configuration file (.ini) accessed by SMT.	None	Void	Public
`getBLEUScore`	This method calculates and displays the BLEU score of a candidate file.	None	Void	Public

`Train`	This method trains the SMT's model through the given source and target text files.	None	Void	Public
`train_append`	This method retrains the default SMT model (norm) through the added entries in the source and target text files.	None	Void	Public
`dictionaryAppend_File`	This method appends the entries of a given file into the dictionary.	None	Void	Public
`dictionaryAppend_Text`	This method appends a given entry into the dictionary.	None	Void	Public



```
if (inputType == 1) {
    System.out.print("\nEnter String: ");
    shortcutText = userInput2.nextLine();
    if (!"".equals(shortcutText)) {
        normalizedText = NormAPI.normalizeDSA_Text(shortcutText);
        System.out.println("Normalized String: " + normalizedText + "\n");
    } else {
        System.out.println("Shortcut String Null.\n");
    }
```

Figure 1: Code for Normalization of Input Text or String



```
} else {
    System.out.print("Enter File Path: ");
    JFrame fr = new JFrame("File Chooser");
    fr.setVisible(true);
    fr.setAlwaysOnTop(true);

    int fileChooser;

    JFileChooser chooser = new JFileChooser();
    fileChooser = chooser.showOpenDialog(fr);

    if (fileChooser == JFileChooser.APPROVE_OPTION) {
        File chosenFile = chooser.getSelectedFile();
        filePath = chosenFile.getPath();
        System.out.println(filePath);
        fr.dispose();
        if (!"".equals(filePath)) {
            NormAPI.normalizeDSA_File(filePath);
        } else {
            System.out.println("Input File Null.\n");
        }
    } else if (fileChooser == JFileChooser.CANCEL_OPTION) {
        System.out.println("Input File selection cancelled.\n");
        fr.dispose();
    }
}
```

Figure 2: Code for Normalization of Input File

For all normalization variations (DSA, SMT, SMTafterDSA, SMTbeforeDSA), the format is similar (**`normalize`**<variation>**`_Text`**) but differ on the approach/es or technique/s used.

3 NormAPI

NormAPI is a Java Application Program Interface (API) used in normalizing shortcut text through the following variations: Statistical Machine Translation (SMT), Dictionary Substitution Approach (DSA), SMT after DSA, and SMT before DSA (default approach). It also provides options for setting the word dictionary, appending entries to the word dictionary, training and retraining new data, setting the Moses configuration file, and calculating BLEU scores. It contains only one class named NormAPI which is located in the normapi source package. The NormAPI class has no super class and no specified properties. In order to use this in other programs, it must be imported (e.g. ‘import normapi.NormAPI;’). The file paths used in the code should not contain spaces to avoid errors. Further details about this class are shown in Table 2 in subsection 3.1 and sample codes for each method are shown in Table 3 subsection 3.2.

3.1 NormAPI Methods

Table 2: NormAPI Class Details

Method Name	Description	Parameters	Result Type	Constraint
`normalize_File`	This method normalizes text from a given file using the default approach of the system.	*String filepath* - file path of the corpus to be normalized	Void	Public
`normalize_Text`	This method normalizes text from the user's input using the default approach of the system.	*String shortcutText* - input text to be normalized	String	Public
`normalizeDSA_File`*	This method normalizes text from a given file using DSA.	*String filepath* - file path of the corpus to be normalized	Void	Public
`normalizeSMT_File**`	This method normalizes text from a given file using SMT.	*String filepath* - file path of the corpus to be normalized	Void	Public
`normalizeSMTafterDSA_File **`	This method normalizes text from a given file using two approaches. It uses DSA first followed by SMT.	*String filepath* - file path of the corpus to be normalized	Void	Public
`normalizeSMTbeforeDSA_File**`	This method normalizes text from a given file using two approaches. It uses SMT first followed by DSA.	*String filepath* - file path of the corpus to be normalized	Void	Public
`normalizeDSA_Text`	This method normalizes text from the user's input using DSA.	*String shortcutText* - input text to be normalized	String	Public

*The file normAPIout (located in C:\cygwin\normapi) is generated by this method.

normalizeSMT_Text	This method normalizes text from the user's input using SMT.	*String shortcutText* - input text to be normalized	String	Public
normalizeSMTafterDSA_Text	This method normalizes text from the user's input using two approaches. It uses DSA first followed by SMT.	*String shortcutText* - input text to be normalized	String	Public
normalizeSMTbeforeDSA_Text	This method normalizes text from the user's input using two approaches. It uses SMT first followed by DSA.	*String shortcutText* - input text to be normalized	String	Public
setDictionary	This method changes the word dictionary accessed by DSA.	*String dictionaryFilePath* - file path of the new word dictionary	Void	Public
setConfigurationFile	This method changes the Moses configuration file (.ini) accessed by SMT.	*String configurationFilePath* - file path of the new Moses configuration	Void	Public
getBLEU	This method calculates the Bilingual Evaluation Understudy (BLEU) score of a candidate file.	*String referenceFilePath* - file path of the reference file	String	Public
		String candidateFilePath - file path of the candidate file		
dictionaryAppend_Text	This method adds a given entry into the dictionary.	*String inputText* - shortcut word placed in the input tag of the Word Dictionary XML file	Void	Public
		String outputText - normalized word placed in the output tag of the Word Dictionary XML file		
dictionaryAppend_File	This method adds a given file's entries into the dictionary. The delimiter used is '-'.	*String dictionaryFilePath* - file path of the text file to be appended in the word dictionary	Void	Public
train	This method trains the SMT's model through the given source (from) and target (to) text files.	*String mainName* - name used for the directory and file names	Void	Public
		String sourceName - name of the source file		
		String sourceFilePath - path		

		of the source file *String targetName* - name of the target file *String targetFilePath*- path of the target file		
createLM	This method creates the language model of a given file.	*String targetFilePath* - path of the target file *String lmFilePath* - target path of the new language model	Void	Public
dictionarySubstitution_File*	This method is the direct function of normalizing a file using DSA.	*String filePathIn* - path of the file to be normalized *String filePathOut* - target path of the normalized output	Void	Public
dictionarySubstitution_Text	This method is the direct function of normalizing a text using DSA.	*String shortcutText* - input text to be normalized	String	Public
statisticalMachineTranslation_File*	This method is the direct function of normalizing a file using SMT.	*String filePathIn* - path of the file to be normalized *boolean first* - indicator if the approach is the first one to be executed	Void	Public
statisticalMachineTranslation_Text	This method is the direct function of normalizing a text using SMT.	*String shortcutText* - input text to be normalized	String	Public

3.2 NormAPI Methods Sample Code

Table 3: NormAPI Methods Sample Usage

Method Name	Sample Code
normalize_File	`NormAPI.normalize_File("C:\\cygwin\\normapi\\in");`

	`String filePath = "C:\\cygwin\\normapi\\in";`
	`NormAPI.normalize_File(filePath);`

	OUTPUT/EFFECT: Using SMT followed by the DSA approach, a file will be created containing the normalized counterpart of the input file.
normalize_Text	`String normalizedText= NormAPI.normalize_Text("nsn k n b");`

	`String shortcutText = "nsn k n b";`
	`String normalizedText =`
	`NormAPI.normalize_Text(shortcutText);`

* The file *normAPIout*(located in C:\cygwin\normapi) is generated by this method.

	OUTPUT/EFFECT: Using SMT followed by the DSA approach, the normalized output of the given shortcut text will then be displayed. “nasaan ka na ba”
normalizeDSA_File	NormAPI.normalizeDSA_File("C:\\cygwin\\normapi\\in");
	String filePath = "C:\\cygwin\\normapi\\in"; NormAPI.normalizeDSA_File(filePath);
	OUTPUT/EFFECT: Using the DSA approach, a file will be created containing the normalized counterpart of the input file.
normalizeSMT_File	NormAPI.normalizeSMT_File("C:\\cygwin\\normapi\\in");
	String filePath = "C:\\cygwin\\normapi\\in"; NormAPI.normalizeSMT_File(filePath);
	OUTPUT/EFFECT: Using the SMT approach, a file will be created containing the normalized counterpart of the input file.
normalizeSMTafterDSA_ File	NormAPI.normalizeSMTafterDSA_File("C:\\cygwin\\normapi \\in");
	String filePath = "C:\\cygwin\\normapi\\in"; NormAPI.normalizeSMTafterDSA_File(filePath);
	OUTPUT/EFFECT: Using DSA followed by the SMT approach, a file will be created containing the normalized counterpart of the input file.
normalizeSMTbeforeDSA _File	NormAPI.normalizeSMTbeforeDSA_File("C:\\cygwin\\normap i\\in");
	String filePath = "C:\\cygwin\\normapi\\in"; NormAPI.normalizeSMTbeforeDSA_File(filePath);
	OUTPUT/EFFECT: Using SMT followed by the DSA approach, a file will be created containing the normalized counterpart of the input file.
normalizeDSA_Text	String normalizedText = NormAPI.normalizeDSA_Text("nsn k n b");
	String shortcutText= "nsn k n b"; String normalizedText = NormAPI.normalizeDSA_Text(shortcutText);
	OUTPUT/EFFECT: Using the DSA approach, the normalized output of the given shortcut text will then be displayed. “nasaan ka ni ba” OR “nasaan ka na ba” OR “nasaan okay ni ba” OR “nasaan okay na ba” OR “nasaan ka ni be” OR “nasaan ka na be” OR “nasaan okay ni be” OR “nasaan okay na be”
normalizeSMT_Text	String normalizedText = NormAPI.normalizeSMT_Text("nsn k n b");
	String shortcutText= "nsn k n b"; String normalizedText = NormAPI.normalizeSMT_Text(shortcutText);
	OUTPUT/EFFECT: Using the SMT approach, the normalized output of the given shortcut text will then be displayed. “nsn ka na ba”
normalizeSMTafterDSA_ Text	String normalizedText = NormAPI.normalizeSMTafterDSA_Text(“nsn k n b”);
	String shortcutText= "nsn k n b"; String normalizedText = NormAPI.normalizeSMTafterDSA_Text(input);

	OUTPUT/EFFECT: Using DSA followed by the SMT approach, the normalized output of the given shortcut text will then be displayed. “nasaan ka ni ba” OR “nasaan ka na ba” OR “nasaan okay ni ba” OR “nasaan okay na ba” OR “nasaan ka ni be” OR “nasaan ka na be” OR “nasaan okay ni be” OR “nasaan okay na be”
normalizeSMTbeforeDSA_Text	String normalizedText= NormAPI.normalizeSMTbeforeDSA_Text(“nsn k n b”);
	String shortcutText= "nsn k n b"; String normalizedText= NormAPI.normalizeSMTbeforeDSA_Text(shortcutText);
	OUTPUT/EFFECT: Using SMT followed by the DSA approach, the normalized output of the given shortcut text will then be displayed. “nasaan ka na ba”
setDictionary	NormAPI.setDictionary(“C:\\cygwin\\normapi\\norm\\dictionary\\WordDictionary.xml”);
	String dictionaryFilePath = “C:\\cygwin\\normapi\\norm\\dictionary\\WordDictionary.xml”; NormAPI.setDictionary(dictionaryFilePath);
	OUTPUT/EFFECT: A new XML file will function as the word dictionary of the system.
setConfigurationFile	NormAPI.setConfigurationFile(“C:\\cygwin\\normapi\\norm\\training\\model\\moses.ini”);

	String configurationFilePath =
	“C:\\cygwin\\normapi\\norm\\training\\model\\moses.ini”);
	NormAPI.setConfigurationFile(configurationFilePath);

	OUTPUT/EFFECT: The configuration file (.ini) accessed by Moses is changed.
getBLEU	String BLEUscore = “”; String referenceFilePath = “C:\\cygwin\\normapi\\norm\\testing\\ReferenceCorpus.txt”; String candidateFilePath = “C:\\cygwin\\normapi\\normAPIout”; BLEUscore = NormAPI.getBLEU(referenceFilePath, candidateFilePath);
	OUTPUT/EFFECT: Displays the BLEU score of the given file.
dictionaryAppend_Text	String inputText = “nsn”; String outputText = “nasaan”; NormAPI.dictionaryAppend_Text(inputText, outputText);
	OUTPUT/EFFECT: A new entry will be converted into XML format and added into the default word dictionary. sn-saan → <entry>sn</entry><output>saan</output>
dictionaryAppend_File	String dictionaryFilePath = “C:\\cygwin\\normapi\\DictionaryEntry.txt”; NormAPI.dictionaryAppend_File(dictionaryFilePath);
	OUTPUT/EFFECT: Entries from the given text file will be converted into XML format and added into the default word dictionary.
train	String mainName = “user”; String sourceName = “shortcut”;

	String sourceFIlePath = “C:\\cygwin\\normapi\\norm\\corpus\\TranslationCorpus.txt”; String targetName = “normalized”; String targetFilePath = “C:\\cygwin\\normapi\\norm\\corpus\\NormalizedTranslationCorpus.txt”; NormAPI.train(mainName, sourceName, sourceFilePath, targetName, targetFilePath);
	OUTPUT/EFFECT: Through the given source and target text files, the model for SMT is trained.
train_append	String sourceFilePath= “C:\\cygwin\\normapi\\source.txt”; String targetFilePath= “C:\\cygwin\\normapi\\target.txt”; NormAPI.train_append(sourceFilePath, targetFilePath);
	OUTPUT/EFFECT: The model for SMT is retrained to include addition training data from the given text file.
createLM	String targetFilePath = “C:\\cygwin\\normapi\\norm\\corpus\\norm.normalized”; String lmFilePath = “C:\\cygwin\\normapi\\norm\\training\\lm\\norm.normalized.lm”; NormAPI.createLM(targetFilePath, lmFilePath);
	OUTPUT/EFFECT: A language model of the given file will be created.
dictionarySubstitution_File	String filePathIn = “C:\\cygwin\\normapi\\in”; String filePathOut= “C:\\cygwin\\normapi\\out”; NormAPI.dictionarySubstitution_File(filePathIn, filePathOut);
	OUTPUT/EFFECT: The input file is normalized using the DSA approach.
dictionarySubstitution_Text	String shortcutText = “nsn k n b”; String normalizedText = NormAPI.dictionarySubstitution_Text(shortcutText);
	OUTPUT/EFFECT: The input text is normalized using the DSA approach. “nasaan ka ni ba” OR “nasaan ka na ba” OR “nasaan okay ni ba” OR “nasaan okay na ba” OR “nasaan ka ni be” OR “nasaan ka na be” OR “nasaan okay ni be” OR “nasaan okay na be”
statisticalMachineTranslation_File	String filePathIn= “C:\\cygwin\\normapi\\in”; NormAPI.statisticalMachineTranslation_File(inputFile, true);
	OUTPUT/EFFECT: The input file is normalized using the SMT approach.
statisticalMachineTranslation_Text	String shortcutText= “nsn k n b”; String normalizedText = NormAPI.statisticalMachineTranslation_Text(shortcutText);
	OUTPUT/EFFECT: The input text is normalized using the SMT approach. “nsn ka na ba”

4 Messages

This section lists system messages (ex. error messages) that the user may encounter while using the system.

4.1 Error Messages

Message	** Parsing error, line <number>, uri file:/C:/cygwin/normapi/norm/dictionary/WordDictionary.xml <System Error> The markup in the document following the root element must be well-formed.
Description	Absence of the opening Word Dictionary tag.
Action	Open WordDictionary.xml then add the opening Word Dictionary tag at the very first line of the file then Save. (e.g. <WordDictionary>)

Message	** Parsing error, line <number>, uri file:/C:/cygwin/normapi/norm/dictionary/WordDictionary.xml <System Error> The end-tag for element type "WordDictio" must end with a '>' delimiter.
Description	Typographical Error within the brackets of the opening Word Dictionary tag; Typographical Error within the brackets of the opening entry tag; Typographical Error within the brackets of the opening input tag; Typographical Error within the brackets of the opening output tag; Missing '>' at the end of the closing entry tag; Missing '>' at the end of the closing input tag; Missing '>' at the end of the closing output tag;
Action	For typographical error: Open WordDictionary.xml, refer to the line number stated in the error message, fix the typographical error then Save. (e.g. <WordDictio> to <WordDictionary>) For missing: Open WordDictionary.xml, refer to the line number stated in the error message or look for the tag with the missing bracket, fix the tag then Save. (e.g. </output to </output>)

Message	** Parsing error, line <number>, uri file:/C:/cygwin/normapi/norm/dictionary/WordDictionary.xml <System Error>

	Element type "WordDictionary" must be followed by either attribute specifications, ">" or "/>".
Description	Missing '>' at the end of the opening Word Dictionary tag; Missing '>' at the end of the opening entry tag; Missing '>' at the end of the opening input tag; Missing '>' at the end of the opening output tag;
Action	Open WordDictionary.xml, refer to the line number stated in the error message or look for the tag with the missing bracket, fix the tag then Save. (e.g <WordDictionary to <WordDictionary>)

Message	** Parsing error, line <number>, uri file:/C:/cygwin/normapi/norm/dictionary/WordDictionary.xml <System Error> The element type "WordDictionary" must be terminated by the matching end-tag "</WordDictionary>".
Description	Absence of the opening entry tag; Absence of the opening input tag; Absence of the opening output tag; Absence of the closing entry tag; Absence of the closing input tag; Absence of the closing output tag; Typographical Error within the brackets of the closing entry tag; Typographical Error within the brackets of the closing input tag; Typographical Error within the brackets of the closing output tag; Typographical Error within the brackets of the closing Word Dictionary tag;
Action	For absence of tags: Open WordDictionary.xml, refer to the line number stated in the error message or look for the tag with the absent tag, add the tag then Save. For typographical errors: Open WordDictionary.xml, refer to the line number stated in the error message, fix the typographical error then Save. (e.g. </WordDictio> to </WordDictionary>)

Message	** Parsing error, line <number>, uri file:/C:/cygwin/normapi/norm/dictionary/WordDictionary.xml <System Error> XML document structures must start and end within the same entity.
Description	Absence of the closing Word Dictionary tag Missing '>' at the end of the closing Word Dictionary tag.
Action	Open WordDictionary.xml then add the closing Word Dictionary tag at the very last line of the file then Save. (e.g. </WordDictionary>)

4.2 Status Messages

Message	Running Dictionary Substitution Approach (DSA) Dictionary Substitution Approach Commencing… Dictionary Substitution Approach Finished!
Description	This message appears whenever the developer uses the DSA Approach. **`normalizeDSA_Text`** and **`normalizeDSA_File`** have the same status message.
Action	N/A

Message	Running Statistical Machine Translation (SMT) Statistical Machine Translation Commencing... Statistical Machine Translation Finished!
Description	This message appears whenever the developer uses the SMT Approach. **`normalizeSMT_Text`** and **`normalizeSMT_File`** have the same status message.
Action	N/A

Message	Running SMT after DSA Dictionary Substitution Approach Commencing... Dictionary Substitution Approach Finished! Statistical Machine Translation Commencing... Statistical Machine Translation Finished!
Description	This message appears whenever the developer uses the SMT after DSA Approach. **`normalizeSMTafterDSA_Text`** and **`normalizeSMTafterDSA_File`** have the same status message.
Action	N/A

Message	Running SMT before DSA Statistical Machine Translation Commencing… Statistical Machine Translation Finished! Dictionary Substitution Approach Commencing... Dictionary Substitution Approach Finished!
Description	This message appears whenever the developer uses the SMT before DSA Approach. **`normalizeSMTbeforeDSA_Text`**, **`normalizeSMTbeforeDSA_File`**, **`normalize_Text`**, and **`normalize_File`** have the same status message.
Action	N/A

Message	Creating directory: <main name>... <main name> has been successfully created!
Description	'main name' refers to the main name for directory and file names. This message appears when the file for training is created. **`train`** uses this status message.
Action	N/A

Message	Copy and Paste the command below on the Cygwin command prompt. (right-click the Cygwin command prompt > click edit > click paste > wait > exit Cygwin) --- <generated command> --- Note: Set the configuration file (Moses.ini) to use the newly trained model! Directory: <NormAPI Directory>\\user\\<main name>\\training\\model\\Moses.ini
Description	This message appears whenever the developer uses the **`train`** function to training data to the system. Also, the cygwin terminal will appear when the function is executed. **`train`** uses this status message.
Action	Copy and Paste the command below on the Cygwin command prompt. (right-click the Cygwin command prompt > click edit > click paste > wait > exit Cygwin)

Message	Creating directory: <main name>... <main name> has been created already!
Description	'main name' refers to the main name for directory and file names. This message appears when there was a file that was created before that has the same name. **`train_append`** uses this status message.
Action	N/A

Message	Copy and Paste the command below on the Cygwin command prompt. (right-click the Cygwin command prompt > click edit > click paste > wait > exit Cygwin)

	-- <generated command> -- <appended characters>
Description	This message appears whenever the developer uses the **`train_append`** function to add more training data to the system. Also, the cygwin terminal will appear when the function is executed. **`train_append`** uses this status message.
Action	Copy and Paste the command below on the Cygwin command prompt. (right-click the Cygwin command prompt > click edit > click paste > wait > exit Cygwin)