Working on existing software systems - LibreOffice

COP701: Software Systems Laboratory
Assignment 3

by
Dharmisha Sharma - 2022MCS2062
Raajita Bhamidipaty - 2022MCS2053

Guide: Prof. Rahul Narain

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Acknowledgement

We would like to thank Prof. Rahul Narain for granting the opportunity of working on this interesting assignment and for his invaluable help and guidance.
A special thanks to Research Centre for Technical Development of Punjabi Language, Literature and Culture, Punjabi University, Patiala for providing Punjabi words data for creation of dictionary and sample of Punjabi wordnet for preparing thesaurus.
We would like to express our deep sense of gratitude to our fellow classmates for their important help and technical suggestions. Finally we would like to thank our families and friends for their constant moral support.

Abstract

LibreOffice is a widely used open source software. It is an open source alternative for MS Word. It supports various languages used across the globe and also supports functionalities like Spell check, thesaurus of a lot of these languages.
Our motive for this project is to add the functionality of our native language i.e. Punjabi by adding features like spell check and thesaurus for our language which is currently not supported by LibreOffice.
In our project we have mainly worked on building a dictionary and thesaurus according to the format that LibreOffice uses and integrating them with the existing software.

Contents

List of Figures

1 About LibreOffice

LibreOffice is an open source alternative for MS Word. Its native file format is Open Document Format (ODF), an open standard format that is being adopted by governments world wide as a required file format for publishing and accepting documents. LibreOffice can also open and save documents in many other formats, including those used by several versions of Microsoft Office.

LibreOffice includes the following components.

1. Word Processor : Writer is a feature-rich tool for creating letters, books, reports, newsletters, brochures, and other documents. You can insert graphics and objects from other components into Writer documents. Writer can export files to HTML, XHTML, XML, Adobe Portable Document Format (PDF), and several versions of Microsoft Word files. It also connects to email client.

2. Calc (spreadsheet) : Calc has all of the advanced analysis, charting, and decision making features expected from a high-end spreadsheet. It includes over 300 functions for financial, statistical, and mathematical operations, among others. The Scenario Manager provides "what if" analysis. Calc generates 2Dand 3D charts, which can be integrated into other LibreOffice documents. You can also open and work with Microsoft Excel workbooks and save them in Excel format. Calc can also export spreadsheets in several formats, including for example Comma Separated Value (CSV), Adobe PDF and HTML formats.

3. Impress (presentations) : Impress provides all the common multimedia presentation tools, such as special effects, animation, and drawing tools. It is integrated with the advanced graphics capabilities of LibreOffice Draw and Math components. Sideshows can be further enhanced using Font work special effects text, as well as sound and video clips. Impress is compatible with Microsoft Power Point file format and can also save your work in numerous graphics formats, including Macro media Flash (SWF).

2 Building LibreOffice

Steps for building LibreOffice on our local machine:

1. Building dependencies: We had to build the dependencies required for building LibreOffice which included python3, gperf, doxygen, autoconf, some dev plugins, etc.

2. Clone the code from sources like: Gerrit, Git, Anongit repository. We cloned the code from gerrit using the following command.
   git clone https://git.libreoffice.org/core

3. Version Compatibility: Make sure the version of all dependencies are compatible with the source code cloned from gerrit.

4. Autogenerate script : After cloning successfully, run the autogen script as follows
   ./autogen.sh

5. Build : Build using the "make" command.

6. Run : Finally we ran the software built on our local system using the following command:
   ./instdir/program/soffice

3 Spell Check in LibreOffice

Hunspell is the spell checker library used by LibreOffice. It provides a system for tokenizing, stemming and spelling in almost any language or alphabet.

Spell checking text consists of the following steps:

1. Parse a document by extracting (tokenizing) words that we want to check
2. Analyze each word by breaking it down in it's root (stemming) and conjugation affix
3. Lookup in a dictionary if the word+affix combination if valid for your language
4. (optional) For incorrect words, suggest corrections by finding similar (correct) words in the dictionary

The "hunspell_check" and "hunspell_suggest" functions can test individual words for correctness, and suggest similar (correct) words that look similar to the given (incorrect) word.

4 Description and Design of Modules used for adding functionality

4.1 High-Level Architecture of Existing Software

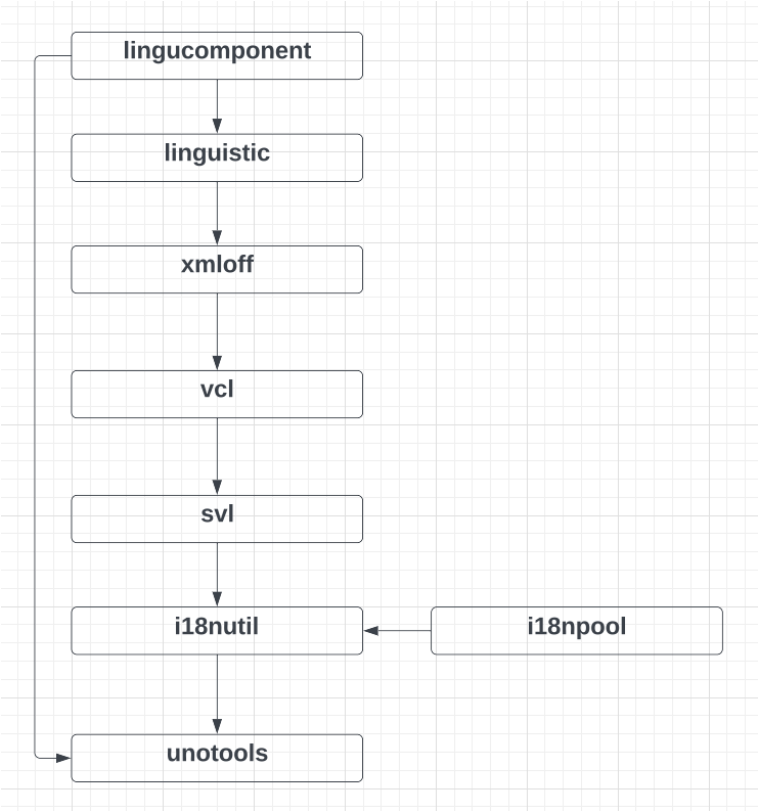

Figure 1: High-Level Architecture

1. lingucomponent: contains spellcheck, hyphenator, thesaurus, etc.
2. linguistic: handles the registered modules for spellchecker, hyphenator and thesaurus.
3. vcl: Visual Class Library (VCL) is responsible for the widgets (windowing, buttons, controls, file-pickers etc.), operating system abstraction, including basic rendering (e.g. the output device).
4. svl: Non-graphic helper code.
5. i18npool: Internationalisation pool (i18npool) framework ensures that the suite is adaptable to the requirements of different native languages, their local settings and customs, etc without source code modification.: helpers for C++ use of UNO.

4.2 UNO packages

UNO packages are UNO components (single libraries or Jar files or more complex zip files that contain one or more libraries| Jar files, type libraries and configuration items), scripts and LibreOffice Basic libraries as zip package. The unotools act as helpers for C++ use of UNO.

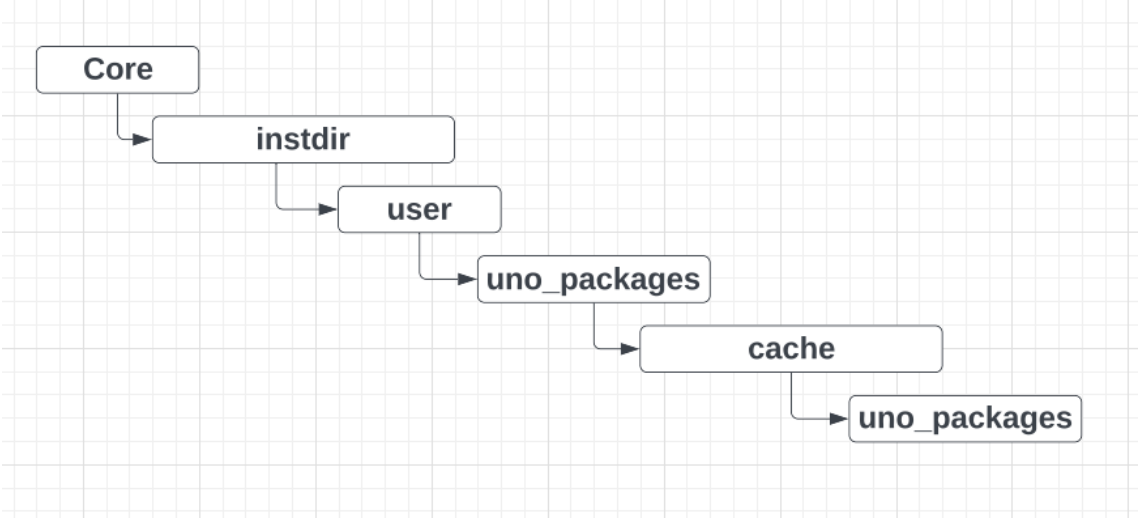

Figure 2: UNO Packages

4.3 Linguistics Components

lingucomponent: contains spellcheck, hyphenator, thesaurus, etc.

4.3.1 Spell Check in lingucomponent

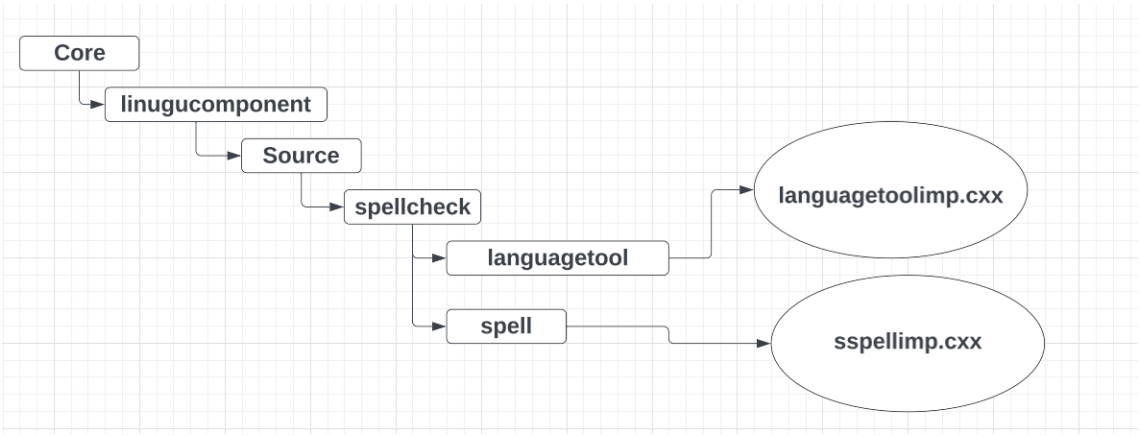

Figure 3: Files for spell checker in lingucomponent

Following is the description of the main C++ code files used for spell check.

1. languagetoolimp.cxx: This file contains a C++ code mainly concerned with the grammatical checks and colour formats of a text. For example, if a word is misspelled, for a grammatical mistake, etc. the colour of line beneath the text is maintained by this file.

2. sspellimp.cxx: This file contains the main C++ code for spell checking. It follows the following steps:
   a. Check dictionary: Currently, one dictionary per language is supported.
   b. Check encoding of dictionary: It should be UTF-8.
   c. Checks both .dic and .aff file for spell check.
   d. Retrieves values when "spell" function is called in case of a misspelled word and gives a list of misspelled words.
   e. It checks .aff file for replacement suggestions of a misspelled word.

4.3.2 Thesaurus in lingucomponent

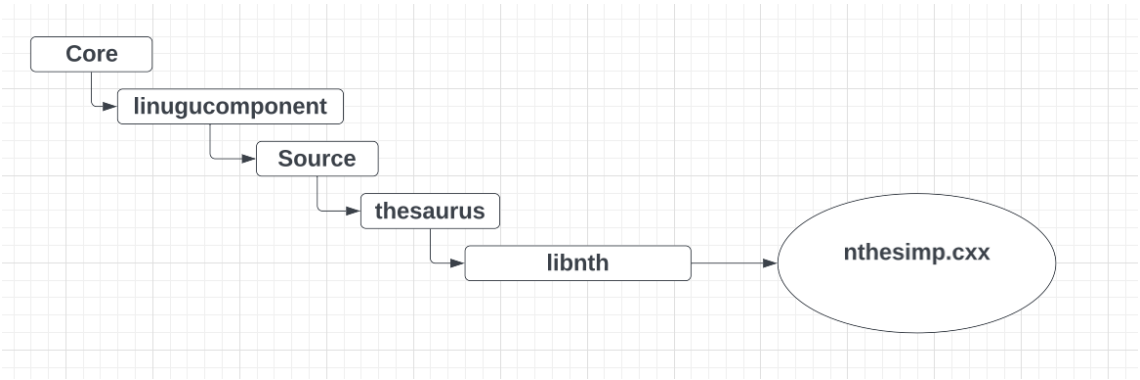

Figure 4: Files for thesaurus in lingucomponent

Following is the description of the main C++ code files used for spell check. 1. nthesimp.cxx: This file contains the main C++ code for the thesarus. It follows the following steps:
a. Gets supported locales from the dictionary in use.
b. Finds thesaurus that matches the locale.
c. Checks encoding: UTF-8.
d. Converts all words to lowercase for searching.
e. Remove category name for affixation and casing.
f. Generate synonyms with affixes.

4.4 Linguistics Components Manager

linguistic module: handles the registered modules for spellchecker, hyphenator and thesaurus.

5 Adding Spell Check

Adding spell check for a local language requires a dictionary for that language. For our project we created

1. META-INF : a folder that contains manifest file
2. dictionary file : pa_IN.dic which consists of words of the language.
3. .aff file (Advanced Forensics Format file): pa_IN.aff which contains replacement rules for spell checking.
4. dictionaries.xcu: which consists of the description of the dictionary and the spell checker.
5. description.xml: description of other files like licenses, information of the version, name of package, etc.

5.1 Dictionary Design

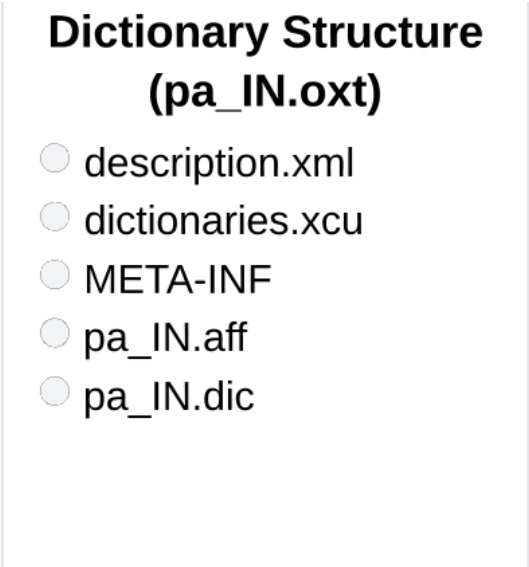

Figure 5: Dictionary Structure

6 Adding Thesaurus

Adding Thesaurus for a local language requires making a synonym file which contains the possible synonyms of all the possible words. For integrating the thesaurus with the existing software we created the following files:

1. META-INF : a folder that contains manifest file
2. dictionaries.xcu: which consists of the description of the dictionary and the spell checker.
3. description.xml: description of other files like licenses, name of package, etc.
4. .aff file (Advanced Forensics Format file): th_pa_IN.aff
5. .dat file : th_pa_IN.dat contains the data for synonyms of the words in the following format:
   word1|WordFormsCount
   (WordForm)|word2|word3|word4

   For example :

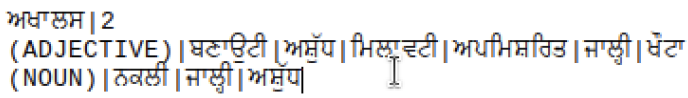

Figure 6: Example of format of a root word with its word form and synonyms

6. .idx file : th_pa_IN.idx is an index files that contains indexes of all the root words present in the thesaurus.

6.1 Thesaurus Design

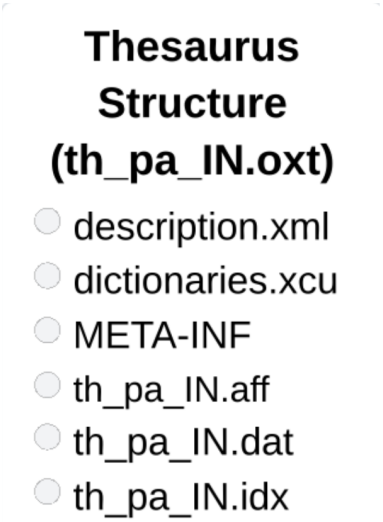

Figure 7: Thesaurus Structure

6.2 Steps for creating Thesaurus

1. We had a Wordnet with words and their possible synonyms. This txt file acted as our base file for creating the thesaurus.

2. We created a module "CreateSynonymsDic.py" which takes as input the txt Wordnet file and gives as output a synonyms.out file which has synonyms in our required format for integrating with the software.

3. Then we created a module "ConsolidateThesaurus.py" which takes a thesaurus structured text data file (synonyms.out) and consolidates the duplicate thesaurus entries based on the word form and outputs a dat file (th_pa_IN.dat).

4. We required an index files which contains the indices corresponding to the words in the .dat file, so we created a module "Dat2IDX.py" which takes as input a .dat file and gives the index file (th_pa_IN.idx).

6.3 Design of Modules for creating Thesaurus

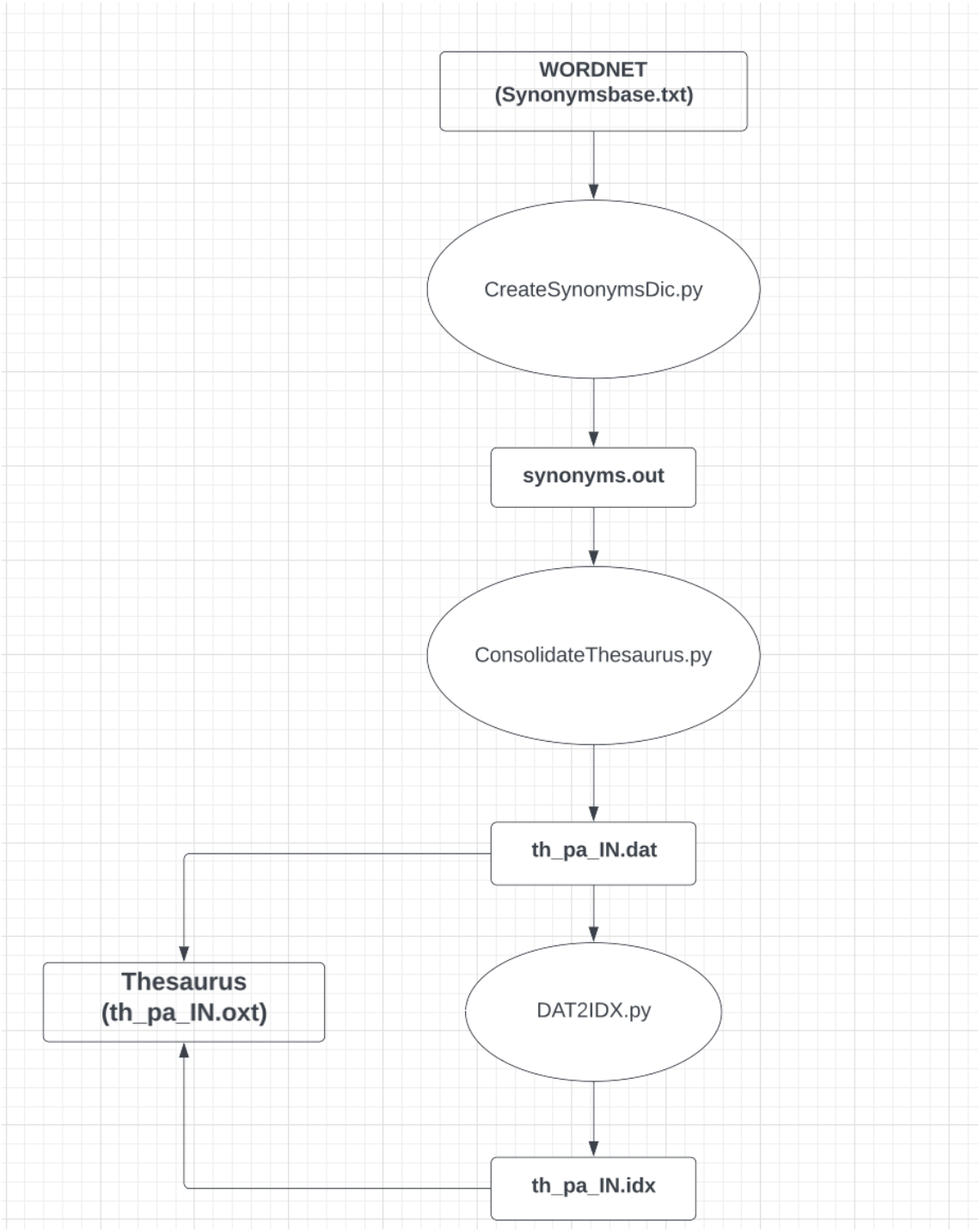

Figure 8: Design of Modules for Creating Thesaurus

7 Integration with original software

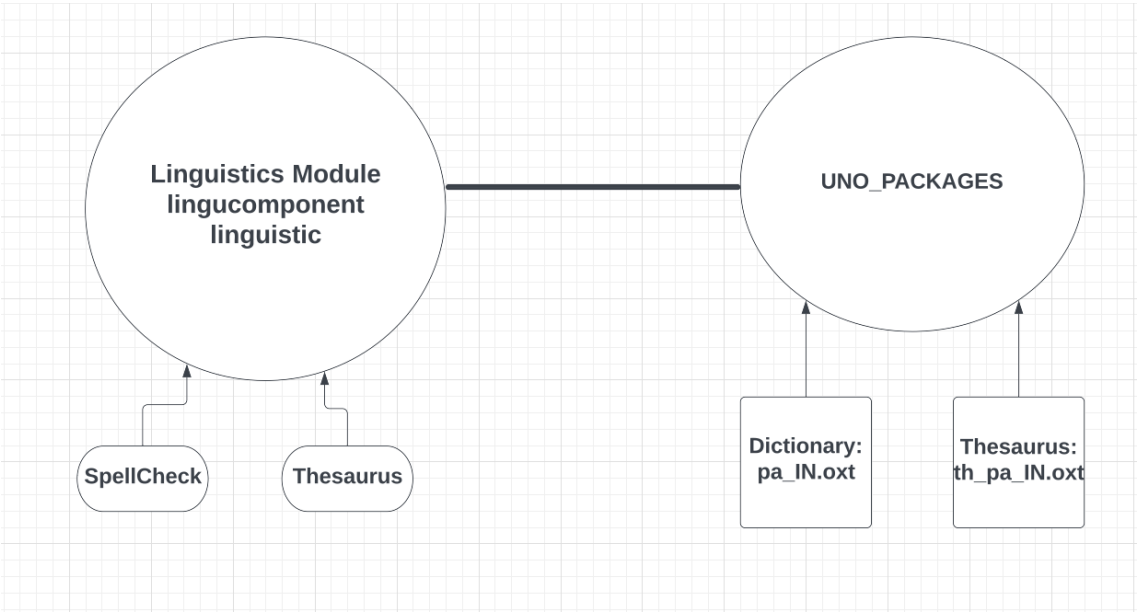

Figure 9: Integration of Dictionary and Thesaurus

The dictionary and thesaurus created are integrated inside the uno_packages which are linked with the linguistic module which contains files for spell check and thesaurus.

8 Screenshots

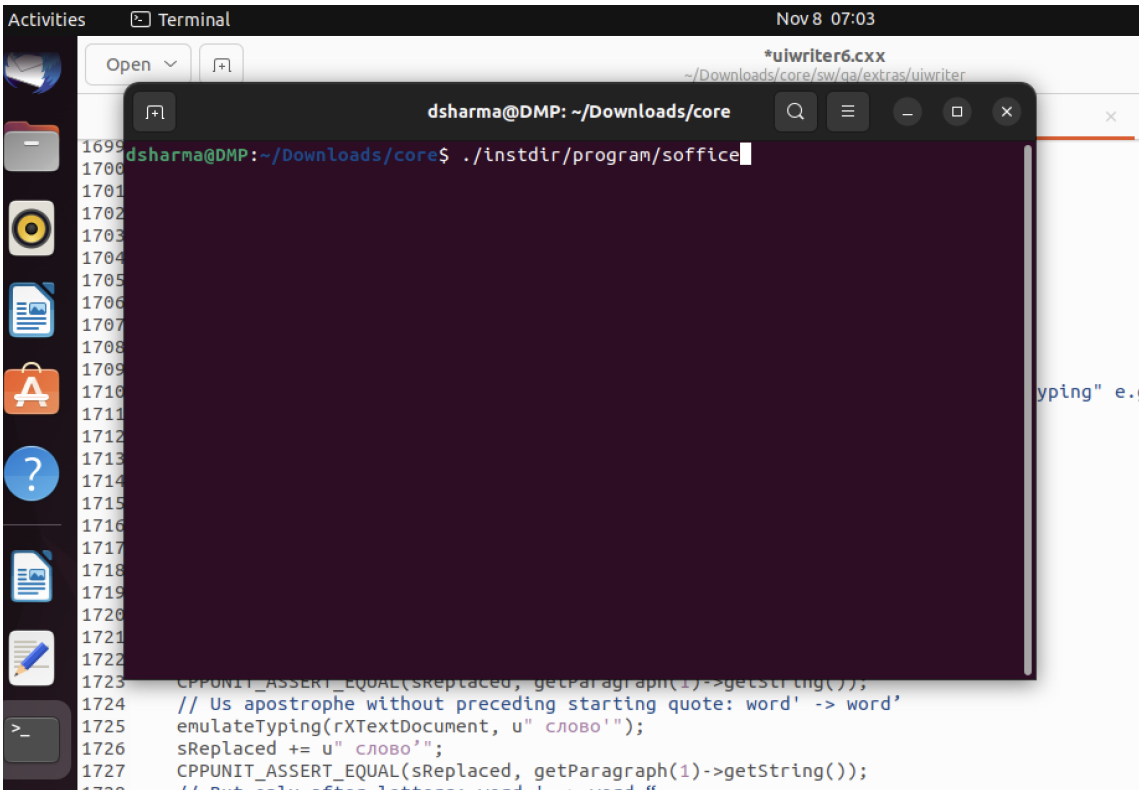

Figure 10: Running LibreOffice Dev

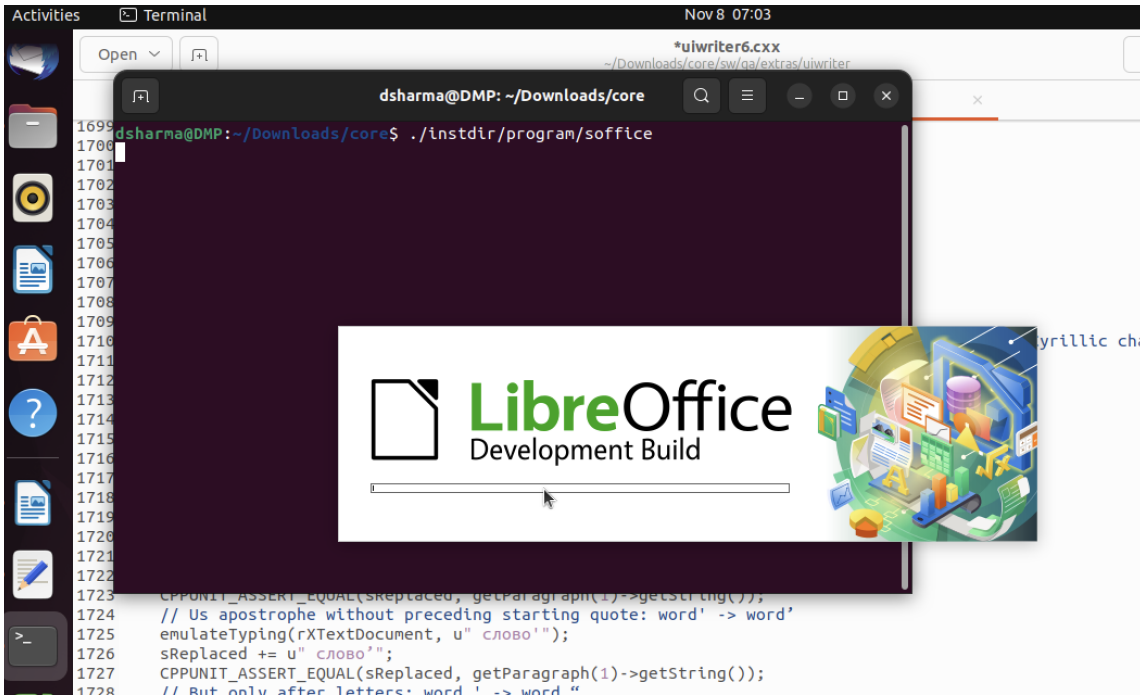

Figure 11: Running LibreOffice Dev

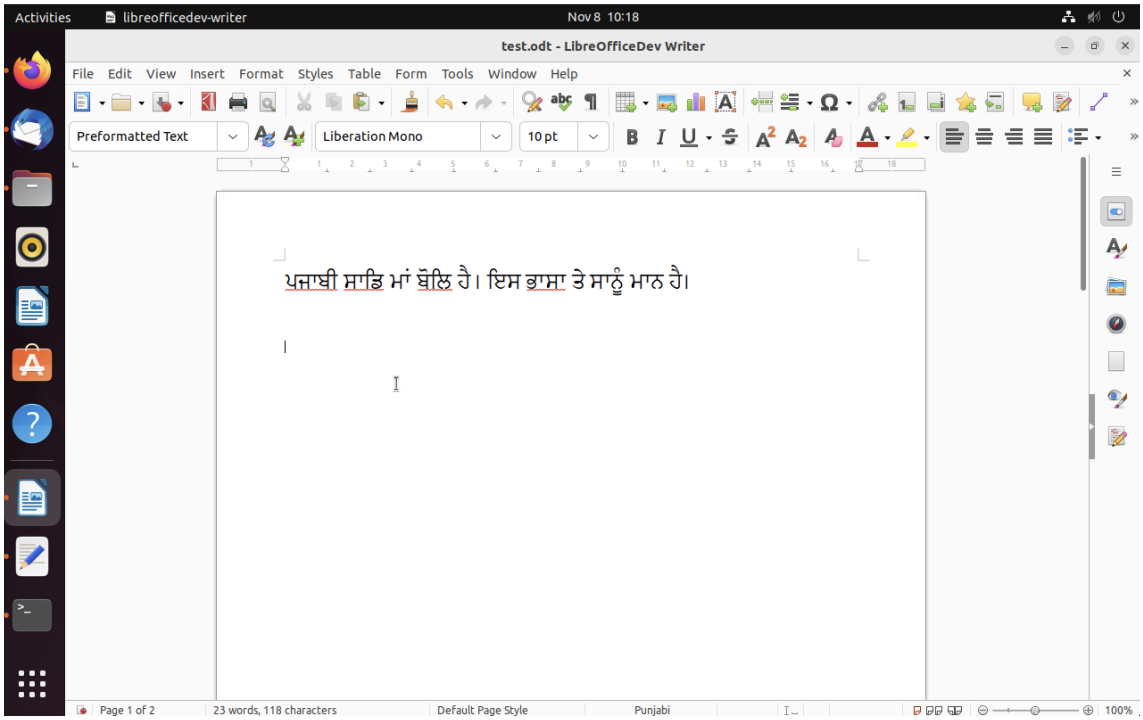

Figure 12: Spell Checker

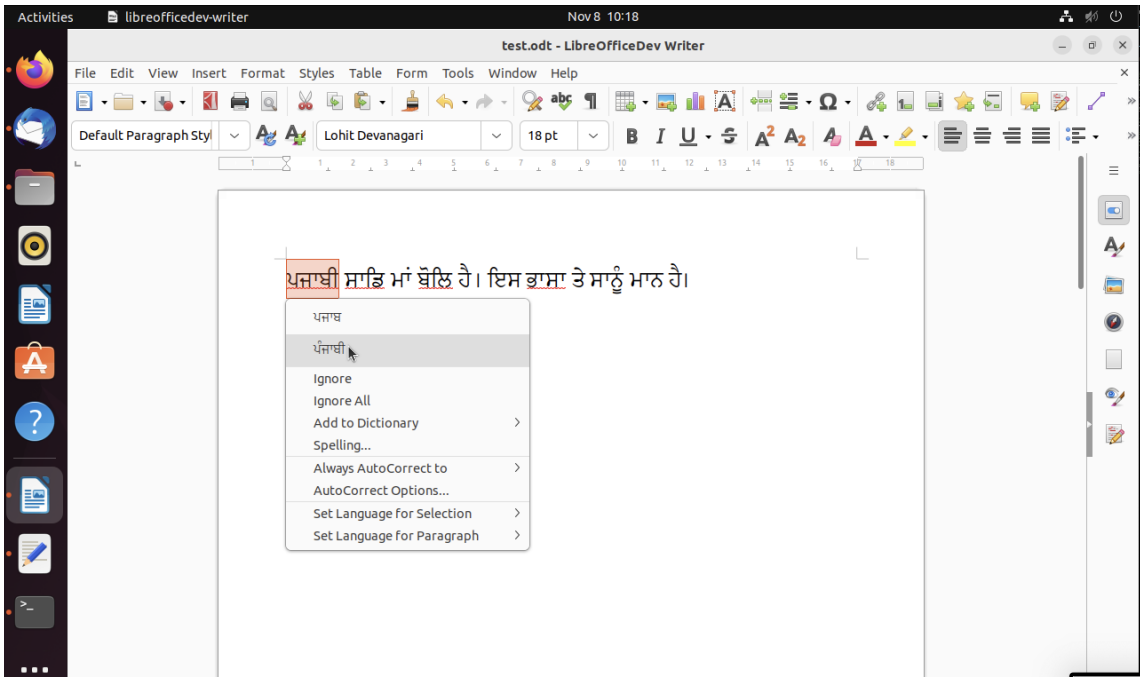

Figure 13: Suggestions by Spell Checker on right clicking on a word

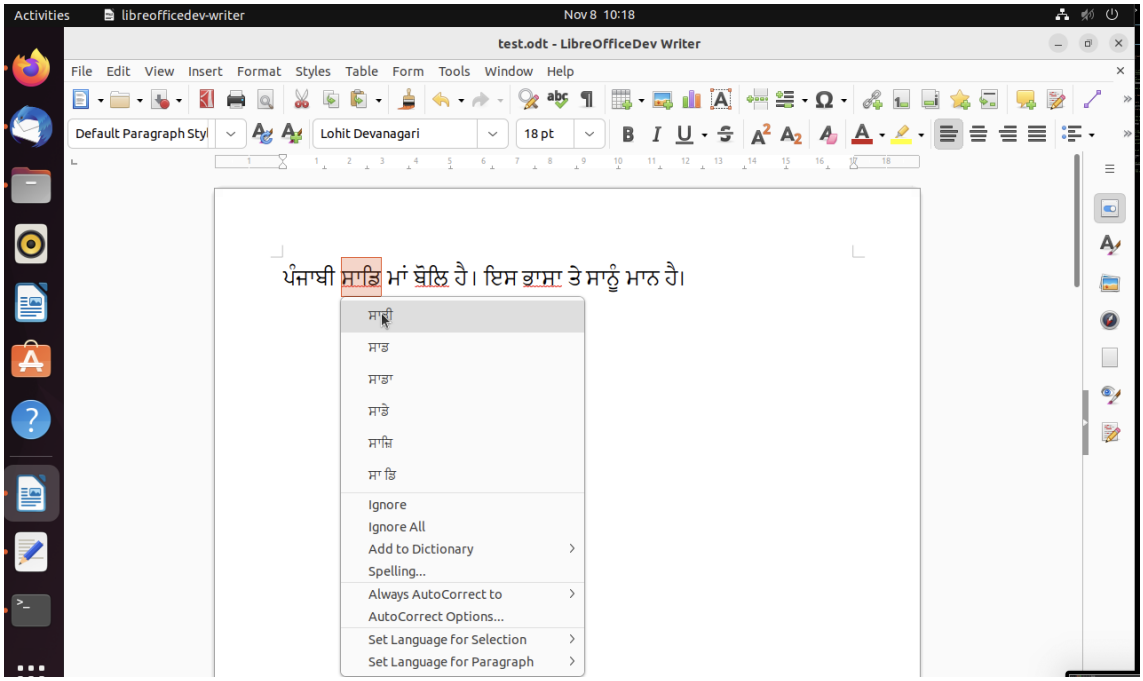

Figure 14: Example of Spell Checker

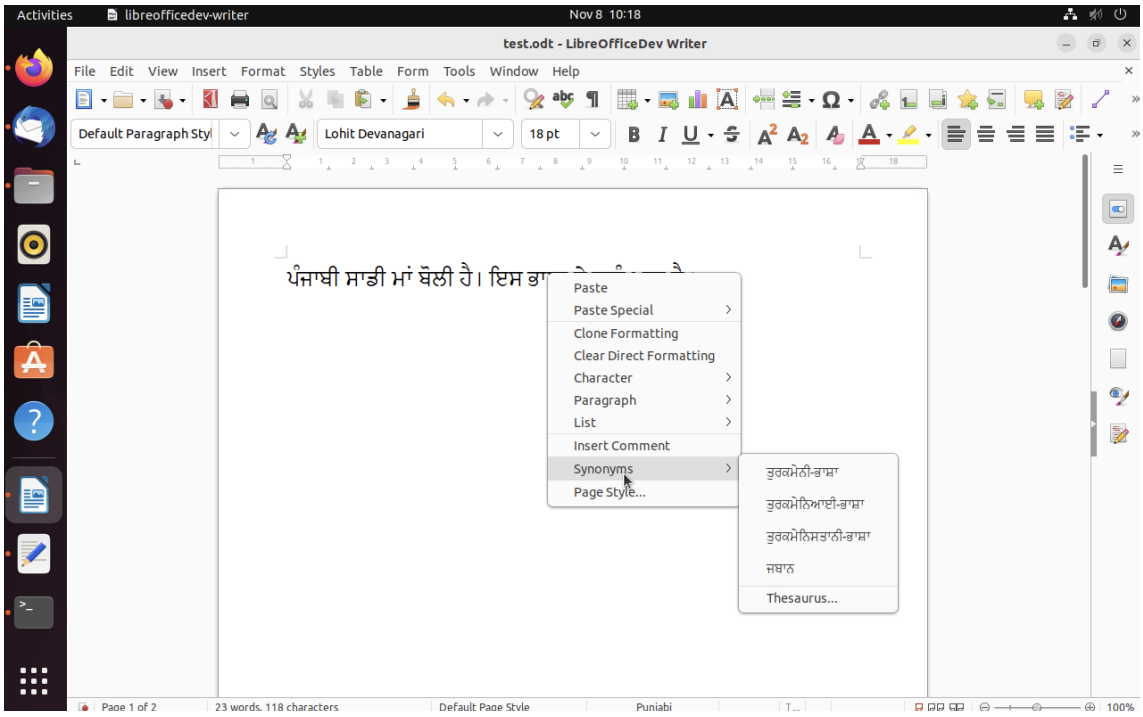

Figure 15: Synonyms Suggested by Thesaurus on Right clicking on a word

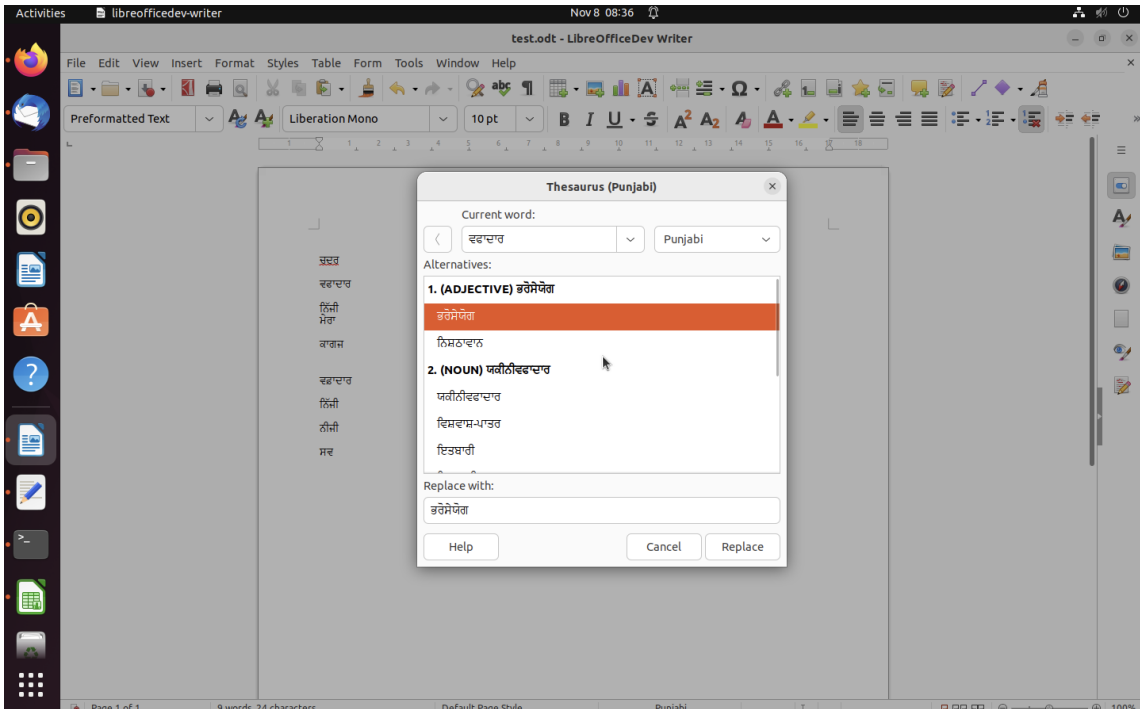

Figure 16: Thesaurus of word according to the form of word(Noun/Adjective/Adverb)

9 Efforts made and Difficulties faced while adding other functionalities

One of the additional functionalities that we required to implement the English to Punjabi dictionary in the LibreOffice was to add an additional option in the context menu that opens when we right click on the word or to add an additional icon in the tool bar that could facilitate the functionality.

LibreOffice uses an internal mechanism, the UNO dispatch framework to dispatch commands from UI to code that handles it. For almost all the commands this code is written in C++. Largely, 3 types of files source files describe UNO commands: .hxx files: contain numeric and symbolic id's .xcu files: Containing definitions of the commands and label .sdi files: contain a list of potential arguments to the command

Implementing a new UNO command requires following comprehensive procedure:

1. Defining the UNO command in specified syntax in configuration (XML) files.
2. Adding the command to relevant GUI files also in XML. Above 2 steps were completed easily.
3. Defining a Slot Id and adding the slot definitions in relevant files. This includes providing info such as type of command (Edit, Insert, Format etc), some flags about flags functionality. Although we were able to locate the files and add in the definitions, we faced some errors due to some understood dependencies.
4. Adding an interface informing what method must be invoked on selecting command in relevant .sdi file.
5. The main files that contain code for executing the command are written in C++ in a switch case format. We must add our implementation to many of the C++ files depending on type of command. Here there was a major problem as it was only allowing us to use a limited number of commands. For implementing said functionality we would have to possibly include some more files and do some reading searching etc. It wasn't allowing us to do that. We weren't able to figure out the internal dependencies very clearly in time due to short span of time. Also, LibreOffice is a humongous repository that contains even legacy code from over 15 years ago. Thus, we found it a little hard to do the analysis.

Another additional functionality we were trying to implement was to perform sentiment analysis on entered text. Although we had the code ready for performing Sentiment Analysis, we couldn't get it up and running due to above problems. Some additional hardships here were, most of the NLP and ML related modules are written in python. The most preferred way for NLP tasks is using modules like TensorFlow, Sklearn, Textbob (used by us) etc. LibreOffice only has support for commands' functionality to be written in C++. This posed a hurdle. Although there are ways around this (using RPC or IPC), both of which require formatting parameters and results in specific ways, the code structure doesn't allow us to add so many dependencies.

We also contacted the developer team of LibreOffice for better understanding of the errors/modifications we were facing, but according to them also since a single module is dependent on a lot of other modules, each feature have many related modules and for adding a feature we'll have to play around those modules. We tried adding UNO commands in the related files of those modules but again since the modules are tightly bound with each other, each command(add, modify, etc) required changes in each of

these modules. Since it is a humongous repository checking each module and making the required changes or checking the errors for each module and rectifying them would require a lot of time.

10 Individual Contributions

2022MCS2062 - Building Dictionary, Thesaurus files in the required format which is compatible with LibreOffice(which include making .dic, .aff, .idx, .xml, .xcu files) and the .oxt folder for integrating Dictionary and Thesaurus with the actual existing software.

2022MCS2053 - Writing the code for Sentiment Analysis, tried adding the additional icon in the toolbar.

11 References

1. https://api.libreoffice.org/docs/tools.html
2. https://github.com/LibreOffice/core
3. https://wiki.documentfoundation.org/
4. https://gist.github.com/Foadsf/58d401c9b9ed5d80f60deee88d1fcdfd
5. https://www.libreoffice.org/about-us/source-code/
6. https://wiki.documentfoundation.org/Development/GenericBuildingHints
7. https://en.wikipedia.org/wiki/LibreOffice
8. https://ask.libreoffice.org/t/multi-language-spell-check/620
9. https://github.com/cpuneet98/Twitter-Sentiment-Emotion-Analysis
10. https://www.cfilt.iitb.ac.in/wordnet/webhwn/