

# IndicWiki Summer Internship - Books

---

[Domain](#)

[Team](#)

[Data Collection](#)

[Sources/ Sites](#)

[Tools used for Data collection](#)

[Images](#)

[Data Storing](#)

[Data Cleaning](#)

[Cases taken care of](#)

[Data Merging](#)

[Version Control](#)

[Sample Article](#)

[Sections](#)

[Jinja template creation](#)

[Edge cases](#)

[Categories, References](#)

[Infobox](#)

[Translation/ Transliteration](#)

[Libraries](#)

[Common Issues while translation, transliteration](#)

[XML Generation](#)

[Quality Review](#)

[Github Structure](#)

## Domain

The domain we worked on was “**Books**”, the aim of the project being generating comprehensive articles for Telugu Wikipedia on 6000+ books, comprising all possible details on a particular book, such as its editions, availability and a lot more, which we have been successfully able to do.

## Team

Member name	Email
Devara Mano Snigdha Reddy	manosnigdhareddy123@gmail.com
Pranav Nandula	npranav.2018@gmail.com
Katta Vishnu Teja	pybeast.0001@gmail.com
Potta Pujitha	p.pujitha11032001@gmail.com

## Data Collection

### Sources/ Sites

Owing to the absence of a unified knowledge base on books, data collected from tens of sources had to be conglomerated, to have a significantly large database, ranging over 6000 books.

***{isbn\_13} used in the links described below refer to the ISBN-13 ID of the book.***

- Google Books API
  - Link : [https://www.googleapis.com/books/v1/volumes?q=isbn:{isbn\\_13}](https://www.googleapis.com/books/v1/volumes?q=isbn:{isbn_13})
  - Format of data available - JSON Webpage
  - Tools used - urllib, json ( Python Libraries )
  - Attributes found
    - ISBN13, ISBN10, Google Books ID ( Primary Keys of the Book )
    - Name, Subtitle, Description, Page Count of the book
    - Author(s), Publisher(s), Originally Published Date of the book
    - Language, Genre(s), Maturity Rating of the book
    - Availability of the book ( as an eBook, ePub and PDF )
    - Thumbnail Links to the Cover Images of the book
  - Note : *Thumbnail Links to Book Covers and the Description of the book had to be eliminated, as images obtained via this source were copyrighted, and the description was not appropriate, because it was either a review, or similar to the summary.*

- Google Books (Website)
  - Link : <https://www.google.co.in/books/>
  - Format of data available - Webpages ( HTML )
  - Tools used - Selenium ( Python Library )
  - Attributes found
    - List of books belonging to the same franchise/series if any
    - List of books published by the same publisher(s) if any
    - List of books written by the same first author if any
    - List of editions of the same book, and details on their ISBN IDs, formats, date of publishing, publisher and page count if any
- Goodreads
  - Link : <https://www.goodreads.com/book/>
  - Format of data available - Webpages ( HTML )
  - Tools used - Selenium, BeautifulSoup ( Python Libraries )
  - Attributes found - Award(s) associated with the book, Ratings, Count of Ratings
- LibraryThing
  - Link : [https://www.librarything.com/search.php?term={isbn\\_13}](https://www.librarything.com/search.php?term={isbn_13})
  - Format of data available - Webpages ( HTML )
  - Tools used - Selenium, BeautifulSoup ( Python Libraries )
  - Attributes found
    - List of characters of the book and summary of the book ( if they exist )
    - Ratings and Count of Ratings ( by users on LibraryThing )
- Amazon
  - Link : [https://www.amazon.in/s?rh=p\\_66%3A{isbn\\_13}](https://www.amazon.in/s?rh=p_66%3A{isbn_13})
  - Format of data available - Webpages ( HTML )
  - Tools used - Selenium, BeautifulSoup ( Python Libraries )
  - Attributes found
    - Dimensions, Ratings and Ratings Count ( by users on Amazon )
    - Details on various formats of the book and their associated purchase/reading links listed on Amazon - Audiobook / Kindle / Paperback / Hardcover, etc.
- Worldcat
  - Link : [https://www.worldcat.org/search?q={isbn\\_13}](https://www.worldcat.org/search?q={isbn_13})
  - Format of data available - Webpages ( HTML )
  - Tools used - Selenium, BeautifulSoup ( Python Libraries )
  - Attributes found
    - A link which displays the availability of a certain book in libraries across the globe
    - Additional details about the first author of the book, such as a link to his/her Wikipedia page, stats such as the number of her/his publications/library holdings/writings/works
    - OCLC Number of the book

- Wikipedia/Wikidata
  - Link : wikipedia.org, wikidata.org
  - Format of data available - Webpages ( HTML )
  - Tools used - Selenium, wptools ( Python Libraries )
  - Attributes found
    - Details about the first author of the book - nationality, alma mater, awards received, notable works
    - Image of the book, author ( taken from Wikimedia Commons )
- OpenLibrary
  - Link : [https://openlibrary.org/isbn/{isbn\\_13}](https://openlibrary.org/isbn/{isbn_13})
  - Format of data available - Webpages ( HTML )
  - Tools used - Selenium
  - Attributes found
    - A link with details on free alternatives to read the book ( eBooks, links on Internet Archive, a site which hosts books for free, etc. )

## Tools used for Data collection

- **Selenium**
  - An automation testing based Python library, primarily used for web scraping
  - **Issues** : No significant issues were encountered in the course of working with this, except for the time delay issue - sometimes, web pages would not load elements on time and that would result in timeouts, exceptions, and end of execution of the program. To avoid this, an additional method was added to keep Selenium webdriver's execution on hold, until some element would appear on the webpage.
- **BeautifulSoup**
  - This was used as an additive to Selenium to navigate through HTML elements and obtain text/information from them, hence, **no major issues** were observed.
- **Wptools**
  - Wptools is a module which when given the search term/query as an argument returns JSON data from the Wikidata page whose title matches the queried term.
  - **Issue** : In most cases, the name of the author/title of the book would not exactly match that on Wikidata because of some abbreviated/additional characters. We used two workarounds to this -
    - Scraping data on the values not found by wptools using Selenium
      - ( this was done for author data )
    - Obtaining data belonging to a particular class ( ex : written work, book ) from Wikidata using SPARQL, merging this data with the data in our DB,

to assign a unique entity ID to matching values in our DB, and then, querying Wikidata using wptools using this entity ID ( since it is the primary key for every entity on Wikidata )

- **urllib**
  - The module “urllib”, along with the module “json” can be used to visit webpages and read data from them, especially if it comprises only raw data, in this case, in .json format. This was used to obtain data from the Google Books API.
  - **Issue** : In many cases, data hosted by the API was incorrect - assigned to the wrong tag, as a result of which data in the required tag would be missing. The only workaround used was manual correction of such inconsistent data.

## Images

- Source : Wikidata/Wikipedia
- **Issue #1** : Since a lot of images weren’t available on Wikimedia Commons, ( the number of book covers found is just 1/3rd the size of the database ), we tried uploading copyrighted, but “permitted for fair use” images of book covers to Wikipedia using Selenium, and another dedicated python module called **pywikibot**. However, this didn’t work, as Wikipedia did not permit using these images and hosting them on Wikimedia Commons.
- **Issue #2** : Due to improper parsing of strings by Selenium, a lot of scraped image titles from Wikimedia Commons turned inaccessible by Wikipedia pages. So to correct these, Selenium had to be re-used to google search all of these images and obtain search results of the Wikimedia Commons page hosting them, to correct them manually in the database.

Parsed Strings	Original Strings
Presentaci <sup>ñ</sup> de cartas credenciales de Miguel Serrano.jpg	Presentaci <sup>ñ</sup> de cartas credenciales de Miguel Serrano.jpg
Slavoj <sup>Žižek</sup> 2015.jpg	Slavoj <sup>Žižek</sup> 2015.jpg
Thomas Andrews <sup>ū</sup> l.jpg	Thomas Andrews <sup>ū</sup> l.jpg

## Data Storing

- **Format #1:** Pickle, .pkl
- Why -
  - .pkl prevents automatic conversion of null values into ‘nan’ and other default data type conversions which usually happens with .csv or .xlsx files, and is hence, the format we’ve preferred to ensure data is kept intact and unadulterated.
- **Format #2:** Comma Separated Value, .csv
- Why -
  - .csv is not a new/unreadable format to computers like .pkl is. Ease of access and editing of these files define why we’ve used .csv files for minor edits.
- **Format #3:** .json
- Why -

- Helps enclose multiple attributes into one attribute as a dictionary, so is useful for in the case of attributes like the “amazon\_formats\_and\_links” attribute in our database which contains details on not all, but only the available formats of a book and its links, which is easier to access via a dictionary when all formats may not exist.

## Data Cleaning

### Cases taken care of

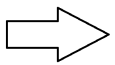
- **Special/Foreign Characters in Values**

- **Issue** : Not all values ( spanning across all attributes ) had pure English characters, and not correcting them to their English equivalent characters would lead to a failure of translation/transliteration. These values were huge in magnitude, however, the only workaround to this was to correct them manually.

```
'\\村上真美子\\': '\\Momiko Murakami\\',
'\\村上龍\\': '\\Ryu Murakami\\',
'\\武蔵・宮本\\': '\\Miamoto Musashi\\',
'\\伊藤潤二\\': '\\Junji Ito\\',
'\\Fedor Mihajlovič Dostoevskij\\': '\\Fyodor Dostoyevsky\\',
'\\Najīb Maḥfūẓ, \\محمفوف, نجيب\\': '\\Naguib Mahfouz\\',
'\\Иван Сергеевич Тургенев\\': '\\Ivan Sergeyevich Turgenev\\',
'\\Honoré de Balzac\\, \\Honoré Balzac\\': '\\Honoré Balzac\\',
'\\Branko·Sou\\閔ek\\': '\\Branko·Soulek\\',
```

- **Very Broadly Defined Book Genres**

- **Issue** : Data on the book’s genre collected was very broad, and was rather a list of all categorical tags of the book. Since such data could not be used as the genre of the book, the workaround deployed was an analysis of 400+ categories existing in the database, normalising them, and fitting them into 12-13 actual “genres”, eight of which are the ones Wikipedia actually defines to be “book genres”.

['Boarding school-fiction']	['German fiction']		<b>Fiction</b>
['Fantasy fiction, English']	['Domestic fiction']		
['Occult fiction']	['Juvenile Fiction']		
['Czech fiction']	['Star trek fiction']		
['Adventure fiction']	['Political fiction']		
['Young Adult Fiction']	['Indic fiction (English)']		
['Fiction']	['Alternative histories (Fiction)']		
['Diary fiction']	['American fiction']		
['Experimental fiction']	['Fantasy fiction']		
['Arabic fiction']	['European fiction']		
	['Australian fiction']		

- **The String-List Anomaly**

- **Issue** : Data enclosed in a list was intended to be collected as a list. However, such data was enclosed in a string containing the list, because data collection in .csv files does not support the list data type. The workaround to this was to use the method **literal\_eval()** from the **ast** module in Python to convert such literals into actual strings, which would however, fail if there were recurring

```
"["Assassin's
Apprentice", 'Royal
Assassin',
"Assassin's Quest"]"
```

↓

```
["Assassin's Apprentice",
"Royal Assassin",
"Assassin's Quest"]
```

The diagram illustrates the conversion of a string containing a list literal into an actual list object. A red arrow points from the string representation to the list representation, with labels "string" and "list" indicating the data types before and after the conversion.

inappropriate ‘, “ or other symbols. Hence, the final workaround was to use this method along with manual removal of apostrophes from such list based string stored attributes.

- **Published Dates vs ‘Originally’ Published Dates**

- **Issue** : A lot of the values of publication date of the book collected were inappropriate because they would correspond to a particular edition of the book, bearing the ISBN used to query sites like Amazon or Google Books to obtain the publication date. Sometimes, these would also be associated with a non printed format/audio format of the book. The workaround to this was twofold -
  - Re-scraping Google Books to obtain the “originally published date” of the book
  - Iterating through all the book editions of a book previously scraped from Google Books and finding the earliest published edition

- **Filling Null Values**

- **Issue** : There were many attributes with highly sparse data, owing to lack of information in the listed sources, such as the first author’s education, nationality, awards received by the book, etc. The best workaround could have been using Selenium to search for such values on Google, but this didn’t work because of multiple dynamic elements in Google search results, so the final workaround was to manually search for such values and fill associated null values in the database.

- **Correcting Inconsistent/Irregular Data**

- **Issue** : Data collected, especially from the Google Books API was occasionally irregular in around a hundred records, owing to irregularity of json data hosted by the Google Books API. For instance, data from languages would appear in thumbnail links, publisher values would appear in the genre, etc. So the workaround to this was to use Selenium to visit HTML Web Pages of the book using the google\_books\_id collected, and access data from there ( since this data is open for manual corrections by editors ). Some of these records also had to be manually corrected.

## Data Merging

- Primary key - isbn\_13
- Process followed, Tool used - Methods from the Python library, Pandas
- FinalKB format - .pkl and .csv
- Final KB rows X columns - 6500 x 73
- [Final KB Link on GitHub](#)
- **Issues** : There were very minor issues observed when attributes serving the same purpose but containing data collected from different sources as values ( and null values )

from each of these were of different formats, for instance, some values contained multiple values as a string, while some values were enclosed in a string, while some null values were denoted as [], etc. So, the only solution to this was to scrutinize all possible values in such attributes, normalize them to a specific format of multiple, singular and null values, and merge them carefully.

## Version Control

- For version control of files used in the project, we used the IIIT GitHub repository, which was especially useful when tasks associated with data cleaning and editing the jinja template had to be performed, when multiple people were working on various versions of the template/data which were sequentially updated.

## Sample Article

- [GitHub Link](#)

## Sections

- **Approach to forming sections**
  - Initially, articles on books from English Wikipedia were scrutinized to identify sections common to most books, and were identified, such as పరిచయం, పుస్తక వివరాలు ( పాత్రలు, కథ ), పురస్కారాలు, మరింత సమాచారం. Most of these sections consist of elementary data about the book and its details.
  - The next set of sections added were based on attributes collected to form the DB, though they weren't really common to all articles, such as రచయిత, రేటింగ్స్, ప్రచురిత పుస్తక వివరాలు, లభ్యత, etc. Most of these sections would focus not exactly on the book, but on subjects associated with it.
- **Sections and their Description**
  - **Introduction (unnamed section)**
    - Contains basic details about the book such as title, authors, publisher, page count, subtitle, language, genre, maturity rating
  - పుస్తక వివరాలు
    - Contains characters and summary of the book (if any) i.e. పాత్రలు, కథాంశం.
  - రచయిత
    - Contains details about the first author of the book i.e. his/her nationality, alma mater, genres of writing, statistics and number of works, awards received and notable works.
  - రేటింగ్స్



- Contains details about the ratings and ratings count of the book as obtained from popular book sales/ eLibrary platforms i.e. Amazon, Goodreads, Google Books, LibraryThing
- పురస్కారాలు
  - Contains a list of the awards and listings associated with the book
- ప్రచురిత పుస్తక వివరాలు, లభ్యత
  - Contains details about
    - ముద్రణలు - Various editions of the book published, each defined in terms of its ISBN13 ID, Published Date, Page Count, Format and Publisher
    - పుస్తక క్రమము - List of books belonging to the same franchise/series if any
    - లభ్యత - Details about the availability of the book and links to different versions on Amazon, Google Books, OpenLibrary and WorldCat if any
- మరింత సమాచారం
  - Contains lists of books published by the same publisher and written by the same (first) author
- మూలాలు
  - Contains a list of all references used in the article

## ● Representational Formats Used

### ● List

Lists have been used to print values of attributes which are multiple in number, except in very few cases. We've used unordered ( bulleted ) lists to print such values, simply to make them more comprehensible. Lists have been used in -

- Characters of the book
- Awards received by the book
- Books published by the same publisher and written by the same first author
- Amazon links to different editions/formats of the same book

### పాత్రలు [\[ edit | edit source \]](#)

- హెన్రీ హెల్మం-బ్రౌన్
- క్లరిస్సా హెల్మం-బ్రౌన్
- సీర్ రోలండ్ దెలాహాయ్
- హుగో బిర్చ్
- జెరెమీ వరెండర్
- మిల్టెడ్ పీకే
- విప్ప హెల్మం-బ్రౌన్<sup>[3]</sup>

### ● Tables

Tables have been used in our articles to render data with multiple entries spanning over five attributes. We've used a sortable table to print entries of book editions of the same book, with each entry containing the edition's ISBN13, Page Count, Publisher, Published Date, Format.

పుస్తక సంచికలు				
ప్రచురణకర్త	ఆకృతి	ప్రచురించిన సంవత్సరం/తేదీ	పేజీ లెక్కింపు	ఐఎస్ బీ ఎన్ సంఖ్య
అల్వర్రోస్టె	పేపర్ బ్యాక్	2015	248	9781444826128
చర్నూవ్	హార్డ్ కవర్	2001	193	9780708992708
ఫ్రెంచ్	పేపర్ బ్యాక్	1956	92	9780573014277
విలియం మోర్ పాపర్ బ్యాక్స్	ఈబుక్	2010	100	9780062006752
సమ్యూల్ ఫ్రెంచ్ త్రాదే	పేపర్ బ్యాక్	2014	142	9780573702334

## Jinja template creation

- [GitHub Link](#)

## Edge cases

- **Making Sure all Edge Cases have been covered**

- The first step was to prevent the rendering of sections/sentences if the attribute the sentence has been constructed on the basis of has a null value for that record. So in the initial stages of designing the template, all occurring null values in the database were identified and these were the very first edge cases added.
- After this, at the end of every iteration of updating the sample article, and subsequently, the template, we tried rendering the article for tens of records, few with dense and few with sparse data. This would lead to the discovery of hidden edge cases which we would then include in the template, and repeat the process.

- **Possible/ Common Edge Cases**

- Null Values
- Printing certain opening sentences in a section only if it would contain data to a certain extent, otherwise, leaving the section empty ( ex : Ratings section )
- Making grammatical corrections to sentences used in the template on the basis of singular/plural value corresponding to the attribute forming it, if the attribute is numerical
- Making grammatical corrections to sentences for low plural values like 2 ( ఇద్దరు ), 3( ముగ్గురు ), 4( నలుగురు ) while letting the rest to be rendered as they are, such as 34 మంది, etc.

• 'గుడ్ రీడ్స్' వెబ్సైటు ఇద్దరు పాఠకుల సమీక్షల ఆధారంగా

• 'గుడ్ రీడ్స్' వెబ్సైటు నలుగురు పాఠకుల సమీక్షల ఆధారంగా

• 'గుడ్ రీడ్స్' వెబ్సైటు 10 మంది పాఠకుల సమీక్షల ఆధారంగా

- **Issues/Improvements Associated with Rendering**

- To improve the readability of values corresponding to certain attributes, we had to render their English versions too in parentheses next to them, for instance - ఈ పుస్తకాన్ని చార్లెస్ ఒస్బోర్న్ ( Charles Osborne ) రచించారు.

- Undesired newline characters would occur in the absence of sentences owing to the values of the associated attributes being null, so these had to be meticulously checked and corrected.
- Sometimes, despite the addition of spaces before punctuation marks like fullstops and commas in the template, they wouldn't get added because of the whitespace elimination formatting used in the template. These also had to be carefully checked and added at all punctuation marks.

## Categories, References

### ● Identifying Categories

- To identify categories for books, we visited existing Wikipedia articles on books to obtain some ideas. On the basis of this observation, we identified the attributes that would define the first two categories : published date of the book, genre.
- However, it was later understood that some infrequent attributes may also be used to categorize these articles. On the basis of suggestions we obtained from our mentors, we also added categories based on awards/listing mentions of the book and its language.
- Some points to note :
  - The categories based on published date, genre and language already exist in Telugu Wikipedia, so we used the same format to create categories based on the same.
  - These categories are generalized across values of all articles, and hence, need not necessarily be existing categories all the time, and can eventually lead to red links too. For example, Telugu Wikipedia has the language based category ఆంగ్ల పుస్తకాలు, but the categories స్పానిష్ పుస్తకాలు or లాటిన్ పుస్తకాలు do not exist, though all of these are based on language.
  - Categories based on awards will be red links, since there is no mention of categories based on book awards in Telugu Wikipedia.

### ● List of Categories

- Category : { published\_date } పుస్తకాలు
- Category : { language } పుస్తకాలు
- Category : { genre } పుస్తకాలు
- Category : {awards\_name} అవార్డు పొందిన పుస్తకాలు / {listing\_name} జాబితాలో చోటు పొందిన పుస్తకాలు

### ● References - the criteria we defined to add them

- Every source from which data on a particular book has been collected has been added as a reference.
- To obtain this reference link, a generalized format of every link was looked up for, such that, if any of the primary keys (ISBN or Google Books ID) is appended to the link, it would redirect to a page containing details of the book, or search

results of that book on the site. For each book, these generalized links with the primary key appended have been added as reference links.

- The reference links appear at the end of the Wikipedia article, listing out all links enclosed in `<ref></ref>`, and these links have been appended at the end of sentences containing data obtained from that source, so as to add a citation. Hence, the reference tags listed at the end of the article are listed only if non-null data has been obtained from that reference link.

## Infobox

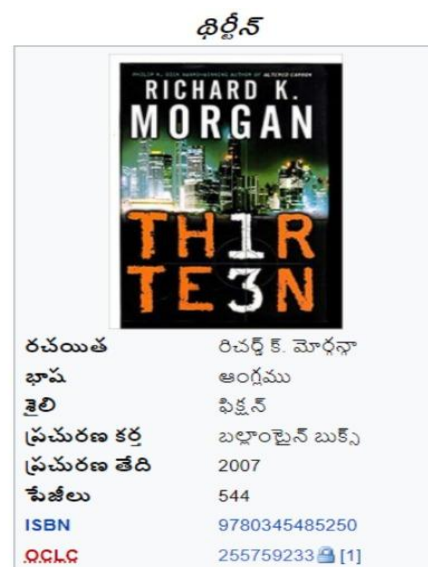
- We had initially planned to make use of the existing Infobox:Books template, but that did not contain a lot of attributes we wished to add to the infobox. Moreover, this template wasn't really suitable for including the image of a book cover into it, so we had to create a custom template with all design specifications like width, headers and data tags.

The infobox's template would begin with the following two parameters -

- **| title**, which in our case is the title of the book, which supersedes all the headers and is rendered on top of the infobox.
- **| image**, which is the name of the book cover image on Wikimedia Commons. Additional attributes like **|image\_size**, **|border**, **|alt** have been added to resize the image, to place a border around the image and an alternate text in case the image doesn't get displayed.

Succeeding these two parameters, is the series of parameters needed to list out the names of attributes, alongside their corresponding values.

- **| author**, for which the value is the list of authors of the book.
- **| language**, for which the value is the language in which the book was written.
- **| genre**, which has the value of the genre the books was written in.
- **| pub\_date**, which has the value as the publication year of the book.
- **| pages**, which has the value as the number of pages the book consists of.
- **| isbn**, which has the value as the ISBN13 number of the book.
- **| oclc**, which has the value, the OCLC number of the book.



The aforementioned three parameters are not dependent on any specific attribute and can be filled with all forms of text, so we were able to use this custom template for the infobox.

# Translation/ Transliteration

## Libraries

- DeepTranslit
  - Used for Transliteration
  - **Issue:** Did not have accurate transliterations of many independent english letters like “to”, “there”, “the”, etc. The workaround was manual correction.
- GoogleTrans
  - Was not used for the final translation, but was experimented with
  - **Issue:** Did not support mass translation ( the limit was just 100 entries of large values like the book’s summary ).
- Bing Translator
  - Used for Translation
  - **Issue:** Would support a translation of upto 2 million characters per account, while the book summary column in our database had more than 4.2 million characters in total. Hence, we had to do this with three different Microsoft Student Partner Azure accounts belonging to IIIT Hyderabad we were given by our mentor.

## Common Issues while translation, transliteration

- **Incorrect Transliteration caused by Singular/Independent English Characters**

- An example of this is “the” getting transliterated to తె or డి while it had to be transliterated to ది or డ . This was observed with “to” ( expected : టు, obtained : తో ), “there”, etc.
- **Workaround** : Manual replacement of such transliterations with their actual Telugu letters.

Original	Corrected
జర్నీ తో తె ఈస్ట్	జర్నీ టు ది ఈస్ట్
తే బూక్ ఆఫ్ ఫైవ్ రింగ్స్	ది బూక్ ఆఫ్ ఫైవ్ రింగ్స్

- **Incorrect Transliteration caused by English data that wasn’t completely suitable for transliteration**

- An example of this is “etc.” getting replaced by ఎత్స్
- **Workaround** : Manual replacement of such transliterations with meaningful words such as మొదలైనవి, ఇతర

- **Incorrect Transliteration of abbreviated words, country names**

- An example of this is “UK” getting transliterated to ఊక్
- **Workaround** :
  - a) **Translating** instead of **Transliterating** such data, if the data consisted of untranslatable words i.e. proper nouns, as any translating module

would identify the difference between words to be abbreviated and words to be actually translated.

- b) Performing manual corrections for values which did not contain proper nouns.

- **Incorrect Transliteration/Translation caused by non English characters**

- As mentioned above in a lot of other sections, transliteration and translation would heavily take a hit because of non English characters used in various values.
- **Workaround** : Manual identification and correction of such values.

## XML Generation

- [GitHub Link](#)

After the template was finalised, the final step was to convert templates of articles in bulk into XML file(s) to be uploaded to tewiki, to publish the articles.

### Procedural Details

Converting templates into XML files involved some key steps like the addition of **<mediawiki>** decorators at various places in the template collection, in abeyance with the syntax prescribed to upload XML files to tewiki, and adding tags of unique identifiers of every article such as **<sha1>** and **<id>** ( on the basis of the range of ids we've been allowed to use ), and other details such as **<timestamp>**, **<contributor>** ( name and id of the user publishing these articles ), and some other tags to complete the technical essentials of the same.

Of all the books in the database, the rendered template of each book has been enclosed between the **<page>** **</page>** tags in the XML File, each "page" inclusive of the aforementioned details such as user details, timestamp, id, etc. This is how the rendered templates of all books in the database have been enclosed and added to the same XML File. All the other tags in the file have been added to comply with the "mediawiki" syntax and enable its linked functionalities/pages when rendered to Tewiki.

### Errors Encountered

The process of generating XML files from the template was pretty straightforward, except for one error that was encountered in due course of doing so - five specific characters used in the template to facilitate the usage of reference tags and other grammatical elements were deemed invalid by the XML parser. These had to be substituted by character representations of the same, but valid in wikitext. The characters and their replacements are as follows -

- **<** ( Less Than )      → "&lt;"
- **>** ( Greater Than )    → "&gt;"

- & ( Ampersand ) → “&”;
- ‘ ( Apostrophe ) → “&apos;”
- “ ( Double Quotes ) → “&quot;”

Therefore, after the conversion of a book’s values into a renderable template, the aforementioned five characters were replaced with their substitutes and then appended to the XML file, and this resulted in perfect rendering of the XML pages generated, with no loss of characters.

### **Eliminating Duplicate Book Titles**

An important step was to ensure there were no duplicates in book titles, in order to avoid XML templates overriding each other, and this is something we’d taken care of in the early preprocessing of the database. However, for a nominal number of duplicate book titles which still remained, the right titles corresponding to ISBNs were queried from secondary sources such as [amazon.com](http://amazon.com), [barnesandnoble.com](http://barnesandnoble.com), [isbndb.com](http://isbndb.com) and were replaced accordingly. Very few records containing duplicate book titles had to be discarded from the dataset.

## **Quality Review**

As part of the quality review of the article, multiple iterations of reviewing were done in order to keep the quality and vocabulary of the article intact. Few instances of the quality review done are as follows -

- **Better Vocabulary**

This was done to replace English versions of words (or) Telugu words with accurate Telugu versions of the same, such as సంచికలు → ముద్రణలు, కథ → కథాంశం, etc. This also comprised of eliminating generalizing plural words like ఎన్నో and కొన్ని to keep statements containing numbers very specific, and substituting words not permitted to be used on Telugu Wikipedia such as మరియు and యొక్క.

- **Using Attributes only of Significant Value ( in terms of content )**

The attributes “maturity rating”, “description” and “bestseller rank” of the book were eliminated, as they were either found to have added no value to the article about the book, or were not appropriate to be added as they were not really informative, but promotional about the book.

- **Improving Readability**

To make values of some attributes comprehensible, such as book awards, the title, authors and publisher of the book, English versions of these values have been added in parentheses next to the Telugu versions of these values, allowing readers to view an accurate representation of the Telugu value in English.

Other attempts towards improving readability were

- Substituting multiple occurrences of pronouns like ఈ పుస్తకం, ఈ రచయిత
- Adding multiple random sentences to existing ones, in order to keep the words used in all articles on a whole as random as distinguishable as possible
- Correcting singular versions of Telugu sentences to make them comprehensible too( i.e those sentences rendered when a value is singular, not plural in number )

- **Keeping the article concise**

Shifting the emphasis from keeping the article populated with the maximum possible word count, to keeping the article concise was one of the objectives of the quality review. For instance, separate sentences “ఈ పుస్తకం ఆంగ్ల భాషలో రచించబడినది.” and “ఈ పుస్తకాన్ని పెంగ్విన్ అనే సంస్థ ప్రచురించింది. ” were combined to make “ఈ పుస్తకాన్ని పెంగ్విన్ అనే సంస్థ ఆంగ్ల భాషలో ప్రచురించింది.”.

Similar corrections were done to keep the article detailed, yet brief - for instance, removing multiple occurrences of references for similar attributes, and multiple occurrences of values like the language of the book.

- **Restructuring Sections to make them informative, yet brief**

The section లభ్యత had too many attributes attached to sentences which were partly unnecessary, since they were all links which would lead to some details of the book. Hence, the entire section was restructured to display links which had a lesser probability of being null values in the database firstly, and then, list out rest of the available values. Similar restructuring was done with the the “Book Series” subsection and the “Characters” sub-section to cut down these lists to a maximum of 10 elements, in order to ensure these subsections wouldn’t be too lengthy.

## Github Structure

Visit our GitHub repository [here](#).

To navigate through the repository, please refer to the below guide.

AUTOXML.xml

Books - Report.pdf

genXML.py

render.py

data

books\_stats.xlsx



<b>AUTOXML.xml</b> - The final XML File Generated
<b>Report</b>
<b>genXML.py, render.py</b> - code used to generate the XML file and render one article respectively
<p>The “data” folder contains</p> <ul style="list-style-type: none"> <li>• A sheet with stats about the data</li> <li>• A sweetviz report on the attributes and values</li> <li>• The final dataset, <b>FINAL-KB.csv</b></li> </ul>
<p>The “<b>raw_files</b>” folder contains</p> <ul style="list-style-type: none"> <li>• raw_code <ul style="list-style-type: none"> <li>◦ All code used to preprocess, scrape, extract data, merge datasets, format data, and code associated with any part of this project</li> </ul> </li> <li>• raw_data <ul style="list-style-type: none"> <li>◦ Versions of datasets used, preprocessed, added more attributes to in the entire project</li> </ul> </li> <li>• raw_templates <ul style="list-style-type: none"> <li>◦ All templates generated and used in due course of the project, and incrementally improved with randomized statements after various quality reviews</li> </ul> </li> </ul>
<p>The “<b>template</b>” folder comprises the final template used to render all articles, “<b>template.j2</b>”, and the Sample Article “<b>Books - Sample Article.pdf</b>”.</p>

