Armenian Books Genre Classification
Spring 2024

Author: Anna Shaljyan
BS in Data Science
American University of Armenia

Supervisor: Natali Gzraryan
MS in Big Data Analytics, IESEG

***Abstract*—This capstone project presents an Armenian Books Genre Classification model designed to predict the genre of a book given details such as the title of the book, author name, description of the book, and name of the publisher. After doing research and identifying available sources, data was scraped from websites of bookstores and online databases. The dataset included 14 features about books belonging to 17 fiction genres. After data preprocessing, methods such as word2vec conversion, imputation of NaN values, Armenian word2vec conversion, and TF-IDF encoding were explored. They were combined with ML algorithms, and several other classifiers, which couldn't handle NaN values. Additionally, LogisticRegression and Feed Forward Neural Network were used. The best model was built using TF-IDF encoding and CatboostClassifier, which handles NaN values and prevents overfitting. The final model had a high multiclass AUC of 90% and accuracy of 65%. Based on this model, a Streamlit app was built, which allows for real-time testing and exploration of the classifier. Overall, this project helps to successfully automate the tedious task of manually understanding the genre of a new book.**

Keywords: Armenian Books, genre classification, word2vec, imputation, Armenian word2vec, TF-IDF encoding, CatboostClassifier, ML algorithms.

I. Literature Review

Nowadays, it is becoming more and more common to use machine learning algorithms when dealing with digital data, or tasks such as classification, predictions and etc. When it comes to text classification, machine learning algorithms such as K-nearest neighbor (KNN), Support Vector Machine (SVM), and Logistic Regression (LR) are often implemented. P. Shiroya, D. Vaghasiya et al. from Parul University in India implemented these techniques when doing books genre categorization using a dataset containing the title and abstract of the book. There is a common issue for librarians, bookstore owners, and readers to categorize the genre of the book, as lots of books lack a genre label, and doing it manually can be a very difficult task [1]. To do so, applying machine learning techniques, researchers from Parul University initially constructed a dataset containing the title of the books, the writer's name, book dialects, their sort and dynamic. Afterward, the data will undergo several stages, such as data pre-processing, text cleaning, feature engineering, training a model, assigning classifiers, and evaluating the results of the model's output. This model can later be utilized as a tool for libraries and bookstores.

Different machine learning algorithms can result in different drawbacks and advantages, so when experimenting with various algorithms, researchers from Parul University found that KNN is costly for computations on large datasets, and it is difficult to find an optimal value of k. However, it is good for classification of unlabeled data and is a common non-parametric method. SVM and LR are easier and more computationally efficient than KN, as SVM can model non-linear decision boundaries, and LR can extend to multiple classes (multinomial regression). Besides advantages, SVM and LR also have drawbacks, such as LR being limited to linear boundaries and requiring data that doesn't contain multicollinearity between independent variables, while SVM lacks transparency in results because of the high dimensionality of text data. To handle these drawbacks and find a good alternative, a combination of several machine learning algorithms is usually used to create a hybrid model, implemented alongside feature selection techniques. It is important to note that not only the right selection of algorithms is crucial, but also good pre-processing of the dataset. Larger datasets decrease the classification error, but if there is redundant or irrelevant data, it will increase the computational complexity and decrease the accuracy of the model [1].

Researchers S. Gupta, S. Jain, and M. Agarwal also did books genre classification using Machine Learning (ML) and Natural Language Processing (NLP) to make the tedious task of manually reading the whole book and then classifying its genre an automated process [2]. A common problem when classifying a book is the huge amount of words that it contains because when using it for building a feature matrix, it can lead to a curse of dimensionality. In such cases, S. Gupta et al. combined similar words into the same class by using NLP and Principal Component Analysis (PCA), which help in feature number reduction. As there are both labeled and unlabeled data about books, labeled data is used for training purposes, and when unlabelled data is added to it, the model's accuracy increases.

The researchers applied the methodology that books are in one-to-one relationship with their genre, as one book can be assigned to only one genre. They only used book titles and their genre without gathering additional data such as publication date, author, etc., combined with the usage of a bag of senses, which allowed them to represent the features. The advantage of bag of senses is that it does dimensionality reduction based on similar terms extracted from Wordnet. Later, TF-IDF is used to compute the weight of each term,

and PCA helps to reduce the dimension of the feature matrix further. In the last step, the AdaBoost classifier is used to do the book genre classification [2].

The flow of the research started with scraping data, and cleaning data by eliminating the most frequent and redundant words that don't affect categorization. When the feature matrix is still sparse, the computation is costly and slow, but with further preprocessing and usage of PCA, the dimensionality will be reduced. The feature matrix is divided into training and testing sets, followed by applying the Decision Tree Classifier and AdaBoost Classifier on the training set to learn from it. After training the model on its training set, the test set is used to make predictions, and the graph plotted showed that only using labeled data gives 81.18% accuracy, while when unlabeled data is combined with labeled data, the accuracy increases to 92.88% [2].

P. Desai, G. Saraiya, and M. Nan from Dharmsinh University in India also saw the problem of books either being badly categorized or defined under a very broad genre of fiction/non-fiction. They proposed to use summaries of the books and apply NLP for classification but didn't limit books to being classified only to one genre [3]. Both labeled and unlabeled data were used, but the volume of unlabeled data was much more. Labeled data was used to train the model, and it was noted that more data combined with similar new data helped to build a better classification model that usually shows as an output the level of accuracy of prediction between 0 and 1. A threshold can be used to dismiss the results that are of low accuracy.

P. Desai et al. scraped data containing book genre, title, and author name, and there were 179 distinct genres, out of which only the most popular genres were left, and then the distribution of each genre was calculated. The data preparation was done on summaries of the books, starting from conversion of text to lowercase, elimination of punctuation marks and stop words, doing word-to-number conversion, and stemming. Additionally, data augmentation was applied to artificially do synonym replacement and increase records in genres with fewer occurrences to have a more balanced distribution. Labeled data containing the genre of the books was mostly categorized among 1 to 4 genres [3].

They used multi-label classification, which allows books to be categorized between one, two, or many labels. They use the F1 score as a metric and create a binary matrix to show 1 next to one of the 139 genres if it is an applicable genre or 0 if it doesn't belong to that genre. Later, TF-IDF vectorizer was used to vectorize the summaries, and after that OneVsRestClassifier was used, which consists of fitting one classifier per class. It has the advantage of being computationally efficient and easily interpretable. To find the accuracy of the developed model, they created a prediction function that, for one dataset, gave an accuracy of 0.81 and for another F1 score of 0.71. They concluded that the Naive Bayes Classifier gives the most accurate predictions, and it is better to categorize books into not one but several genres for better understanding [3].

II. Introduction

The initial step was to do research to understand whether there were available resources to use immediately from the web or whether additional actions were required. There were many available datasets on Kaggle, Github, or other platforms on books, but most of them contained only a few features and were based on English books. As the project's aim was to build an Armenian books genre classifier, none of these datasets was suitable for the goal. We decided that the right approach was to concentrate on web scraping and composing a list of Armenian websites, either of libraries or bookstores, that contained information on Armenian books or books translated into Armenian. After composing a list, we also identified what features were present in the data of each of the bookstores and libraries. As a result, nine primary sources were identified, resulting in approximately a database of 15000 books. However, as several of these websites contained information in English or Russian rather than Armenian, few of these were not desired for scraping. Another limitation was category mismatch on websites, especially bookstores, as one website concentrated on very broad categories and another on very detailed separation. To minimize category mismatch as much as possible, we decided to concentrate on scraping fiction books with their respective categories, as non-fiction book genres were too broad and too many.

III. Data Scraping

A. Goodreads data

The first source was **Goodreads**, as it is the world's largest website for readers. It helps readers discover new books in all kinds of genres and is a great tool for keeping track of reading progress and goals. Goodreads has a shelf of Armenian books containing very detailed information about 800 books such as title, author name, category labels, description of the book, number of pages, average rating, number of ratings and reviews, ISBN of the books, name of the publisher, year of publication and much more. However, there were exceptions, and some of the books didn't have various details.

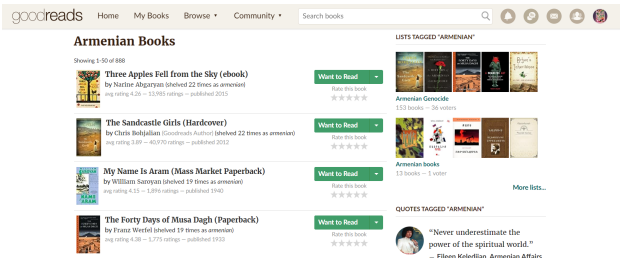

Fig. 1: Example of how information about books looks in Goodreads *"Armenian Books"* shelf

B. Zangak, Bookinist, and grqaser.org websites

Other sources were **Zangak** and **Bookinist** bookstores' websites, which contained both fiction and non-fiction books

displayed in Armenian, English, and Russian. Zangak bookstore's website was well structured, so we scraped data about 1260 fictional books under several genres, containing information such as the title of the book, author name, description of the book, price of the books, and several other additional parameters.

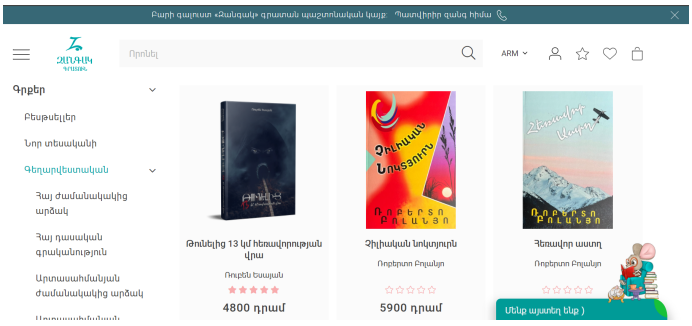

Fig. 2: Example of how information about books looks on Zangak bookstore's website and what features are present

For scraping Zangak's data, we used **Selenium Webdriver**, which allows us to automate the process of obtaining the required information. The important portion of the work was to do research and understand the HTML structure of the webpage to identify patterns for the selectors that were desired for scraping. The desired features were the title of the book, name of the author, price of the book, description of the book, and content under the "*More Info*" section if it was applicable. We identified the genres that we want to gather data on and their base links, so later, we can use "*xpaths*" for each of the features to start the scraping process. We looped through each URL of corresponding genres and scraped information on different features present for each book. We saved the checkpoints for each genre, and the bookstore's concatenated full dataset was saved as a separate Excel file.

However, when we started to scrape Bookinist's website, we noticed that it was poorly structured, and for fictional books, there were four broad categories matching Zangak's defined genres, which were Armenian classic literature, Armenian contemporary literature, Foreign classic literature, and Foreign contemporary literature. Because of the website's poor structure, we decided to use a Chrome Extension called **Webscraper.io** [4], which is a dynamic automated scraper where you build a scraper Site Map giving attribute names and structure examples for different selectors, so you can scrape the website and save the output either as CSV or Excel file.

We used this technique for **grqaser.org** too, which contained more detailed descriptions and genre classification for each book, resulting in a database of 866 books. Using Websraper.io helped to fasten the process of scraping because of its pagination handlers and scroll-down feature. At the end, the database containing information about books summed up to approximately 4000 titles.

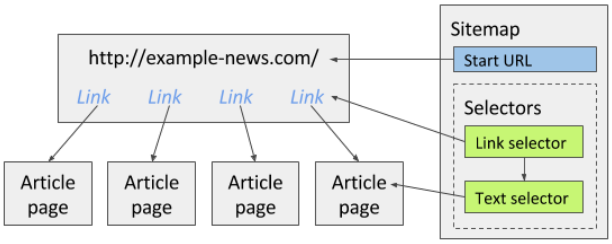

Fig. 3: Example Sitemap structure from Webscraper.io's documentation

IV. Data Preprocessing

A. Cosine similarity calculation

After scraping the Goodreads dataset, we identified that each book contained almost 100 labels, but we needed just one label matching the most accurate genre. We decided to establish a list containing 20 popular book genres, which would serve as a base for further steps:

- Mystery
- Thriller
- Romance
- Science Fiction
- Fantasy
- Historical Fiction
- Horror
- Crime
- Biography
- Memoir
- Self-Help
- Young Adult
- Children's Literature
- Adventure
- Contemporary
- Dystopian
- Suspense
- Poetry
- Comedy
- Drama

As labels contained textual data and the project's methodology was to use NLP techniques, we used spaCy, which is a free, open-source library in Python and has many features such as Named Entity Recognition (NER), Part-of-Speech tagging (POS), text classification, morphological analysis, linguistically-motivated tokenization and much more. We initially loaded spaCy's English tokenizer, preprocessed and cleaned labels of books along with a defined popular book genre list, and then proceeded with word2vec transformation again using spaCy. The last step in this process was to calculate cosine similarity (a measure in data analysis that calculates the similarity of two non-zero vectors), which will iterate over the popular book genre list and labels list for each book and do the comparisons by giving a high score to the genre that is the most similar to one of the genres from the base list. Initially, our threshold was strict at 0.9, but we noticed that such a high threshold eliminates a genre label for lots of books, so we reduced the threshold to 0.75, and for each book, only one matching genre was left.

B. Regex and data types standardization

As several sources for data gathering were used, features of books contained data in different formats, and standardization was required. It was important to normalize data types of

features to be able to later use them accurately. Initially, functions such as numeric value extraction and textual data extraction were defined to change column types to object or Int64. For example, the numeric value extraction function helped to change the number of reviews from 100250 *Reviews* to 10250. Otherwise, the data type of *"Number of reviews"* column would be an object, and we wouldn't be able to use it for analysis. The text extraction function helped to omit numbers in brackets when the desired data type for the column was object. Regex was used to specify patterns in text in columns that contained non-numeric characters or additional information in brackets, which also helped to normalize the data types.

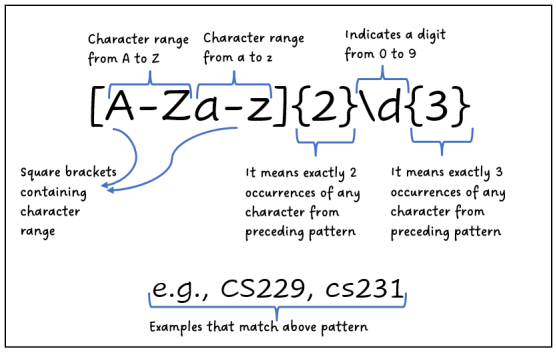

Fig. 4: Regex common patterns

As a result, the dataset contained 14 columns belonging to 3 different data types: object, Int64, float16

```
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Price               1257 non-null   Int64
 1   Genre               6640 non-null   object
 2   Pages               271 non-null    Int64
 3   Avg. Rating         2719 non-null   float64
 4   Number of Ratings   2719 non-null   Int64
 5   Number of reviews   2719 non-null   Int64
 6   Year of Publishing  332 non-null    Int64
 7   Age group           329 non-null    Int64
 8   Title_Vec_0         6640 non-null   float32
 9   Author_Vec_0        6640 non-null   float32
 10  Description_Vec_0   6640 non-null   float32
 11  Publisher_Vec_0     6640 non-null   float32
 12  Reader_Vec_0        6640 non-null   float32
 13  Language_Vec_0      6640 non-null   float32
dtypes: Int64(6), float32(6), float64(1), object(1)
```

Fig. 5: Data types of the resulted dataframe

C. Data translation

Another important issue that required handling was the language of scraped data in the case of Goodreads. Even though the data scraped was about Armenian books, the language used for all the features was English. As our model was going to work on Armenian text, we needed to find a method to translate the data available from English to Armenian as accurately as possible so as not to lose much semantic meaning. To be able to translate the present data in Goodread's dataset, we initially preprocessed the text and removed special characters and symbols. Then, we chained several translation Python libraries to change from one method to another in case of failure. We used packages such as **detect** from the langdetect library, **Translator** from the googletrans library, and **MyMemoryTranslator** combined with **GoogleTranslator** from the deep_translator library. We changed several methods to use packages that were free of cost and not computationally costly to process on GPU. This technique helped to preserve most of the scraped data from the Goodreads website, as we eliminated the poor outcomes. In the end, our dataset contained 14 features such as the title of the book, author name, price of the book, description, genre label, publisher name, number of pages, name of the reader (in case of audiobooks), language of the books and numeric data such as average ratings, number of ratings and reviews, year of publishing and age group suitable for reading the mentioned books.

V. Methods

A. Word2vec and Imputation

After data preprocessing, our data contained 6 columns of type Int64, 1 column of type float64, and 7 columns of data type object. To be able to build models using machine learning algorithms such as **RandomForestClassifier**, **DecisionTreeClassifier**, **LogisticRegression**, and **MLPClassifier**, which were suitable for our classification task, we needed to handle columns of object data type. However, as these ML algorithms don't support NaN values, there was another problem of handling missing values in non-target columns. To be able to use columns of object type while keeping the semantic meaning, we tried **word2vec** [5] conversion, which trains word2vec models for each text column, vectorizes the textual data, or returns NaNs when words are out-of-vocabulary and then creates separate columns for each element of the vector. For handling NaNs, we tried imputation, which supports several strategies, such as replacing NaNs with mean, median, most frequent, or constant values. Unfortunately, only a few columns, such as Title, Author, and Description, contained not many non-null rows, so this was not a good technique for handling NaNs, and it affected the accuracy of tested models.

When using word2vec conversion, we also noticed that our dataset is heavily imbalanced because we have 30 book genres, where the upper portion contains many book examples, while some genres have as few as 1-10 book examples. We decided to merge child genres with their parent genres so the problem of imbalance would be minimized, and we would also decrease the number of available genres. It helped to increase accuracy a little bit, but still, the best was 60% accuracy and a low multiclass AUC score. Besides the above-mentioned methods, we also tried **BaggingClassifier**, **StackingClassifier**, **VotingClassifier**, and building a custom Feed Forward Neural Network.

The Classifiers didn't differ much from previous results, but when we built FFNN, we understood that as our dataset

Classifier	Accuracy
RandomForestClassifier	0.5542271562766866
DecisionTreeClassifier	0.5123825789923142
LogisticRegression	0.6302305721605466
MLPClassifier	0.7386848847139197
BaggingClassifier	0.5609939759036144
StackingClassifier	0.6129518072289156
VotingClassifier	0.5572289156626506

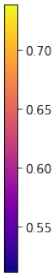

Fig. 6: Different methods with respective accuracies

is not very big, it is not suitable for this task, and even hyperparameter tuning with **RandomizedSearchCV** won't help to set up such parameters that our model will give good accuracy.

B. Armenian word2vec conversion

As word2vec from the gensim library was trained on an English language dataset, we decided to try the Armenian word2vec library downloaded from the GitHub repository of **YerevanNN** [6], which was trained on Armenian Wikipedia. To not deal again with NA handling, we decided to use **CatboostClassifier**, which is an open-source boosting library developed by **Yandex** and is suitable for tasks such as classification where many independent features are present. **Catboost** has built-in methods for handling missing values without the need for imputation. It is robust to overfitting, supports GPU version for faster training, and there is no need for extensive parameter tuning [7]. Even though it had many useful features matching our objectives, combined with Armenian word2vec conversion, it didn't give high accuracy and multiclass AUC scores.

C. Further dataset expanding

Before trying another approach, we decided to fix the data imbalance issue as much as we could and scraped additional data for popular categories romance, horror, fantasy, and mystery & thriller, resulting in a dataset of approximately 7000 books. We again used several translation libraries to translate the gathered data from Goodreads into Armenian. When we rerun the codes for word2vec and Armenian word2vec methods, we noticed that data is prone to overfitting and decided to explore the issue by printing the genres that are mostly incorrectly labeled. As a result, we identified that our models mostly find it difficult to identify between horror and mystery & thriller, Armenian classic literature and Foreign Classic Literature, Armenian Contemporary Literature and Foreign Contemporary Literature, and several other cases, so we did additional merges. After a couple of more merges, our dataset now contained 17 distinct genres, and data imbalance was less noticeable.

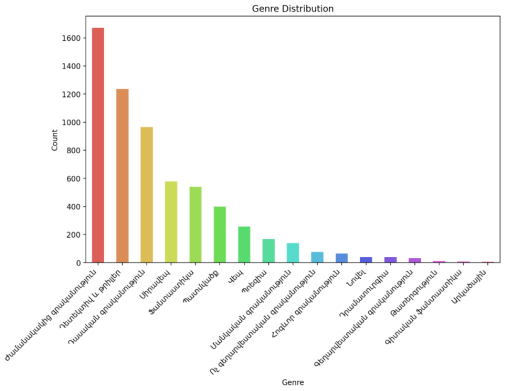

Fig. 7: Genre distribution of modified dataset

VI. Results and Discussion

A. TF-IDF encoding

From the word2vec package of the gensim library and Armenian word2vec, we switched to **TF-IDF** [8] encoding, which is short for Term Frequency Inverse Document Frequency of records and helps to calculate how relevant a corpus is to a text. It follows the logic that the weight of a term proportionally increases to its frequency and is a widely used statistical method in NLP and information retrieval. So, after data preprocessing, we filled the NaN values with empty strings in our desired feature columns and started the TF-IDF encoding by first combining these columns into a single text column and then applying TF-IDF encoding by initializing TfidfVectorizer(). As a result, we create a TF-IDF matrix by fitting the initializer tfidf_vectorizer to our combined single text column and start the train/test splitting by contributing 80% to the train set and 20% to the test set. Besides that, we also fix a random state so our code is reproducible and doesn't split the data differently after each rerun.

B. Catboost model experementations

After applying TF-IDF encoding, we experimented with CatboostClassifier and tried different values for parameters such as the number of iterations, learning rate, depth, L2 regularization extent, and so on. Our target evaluation metric was **Multiclass AUC**, as it is the most suitable metric for evaluating the results of our classifier model, and as we trained our model on GPU, it sped up the process significantly. Increasing the number of iterations and depth of the model combined with a high border count helped to capture more complex patterns, while increasing the L2 regularization level helped to manage overfitting and get correct evaluation results. We also played with the learning rate and saved the results of each such experiment as catboost model and trained TF-IDF encoding matrix *.pkl* files.

C. Best Catboost model

As the best model, we saved the Catboost model which had $\approx 65\%$ accuracy and $\approx 90\%$ multiclass AUC. Higher accuracy

and AUC scores indicated that data expansion and merges of similar genres helped our model to predict better for unseen data. Additionally, experimenting with parameters and defining higher number of iterations or depth of the tree, resulted in model's understanding of more complex patterns. However, as there was still an imbalance in our dataset, and it contained less than 10000 titles, the accuracy couldn't have been much higher. Additional data expansion and preprocessing would have been required to achieve higher results. If there were resources such as stronger GPU or paid translation API, the whole flow of building models would become faster.

D. Demo with Streamlit app

Then, we used the final model's catboost and TF-IDF encoding .pkl files to build a **Streamlit** [9] app for demonstration purposes. This Streamlit app supports real-time experimenting with our model, where the user can give the title of the book, the description of the book, the name of the author and publisher, and see the predicted top 5 genres with their respective probabilities. It should be noted that it is optional to fill in information on the description of the book, author name, or publisher name. However, the more information our model is given, the easier it becomes for it to predict the accurate genre.

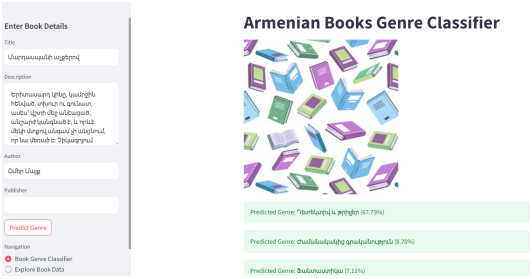

Fig. 8: Output of model's prediction in Streamlit app with respective probabilities

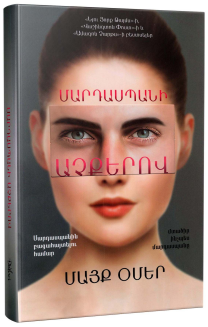

Fig. 9: Cover of the given book [10]

Besides the experimenting page, our app also has the option of exploring our dataset, where, with the help of bar chart visualization, we display the frequency of each genre in our dataset. Additional visualizations that portray additional information about our dataset and results might be added in the future. The users can also view our dataset and filter it by genre.

Fig. 10: **Explore Books Data** page on the Streamlit app

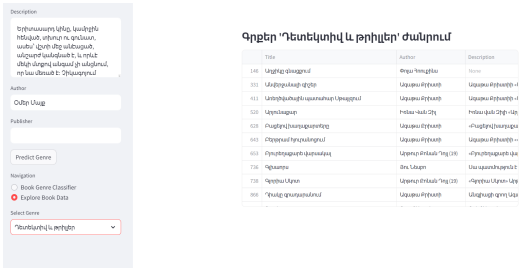

Fig. 11: Example of filtering our dataset based on a genre and viewing details about displayed books

VII. Conclusion and Future Work

This project has developed an **Armenian Books Genre Classification model** to predict the genre of a book given details such as the title of the book, description of the book, author, and publisher name. The model was built using **TF-IDF** encoding to transform text columns of object type to vector format and fit them to a model for classification. Other methods besides TF-IDF were used, such as the word2vec package from the gensim library and the Armenian word2vec package developed by YerevaNN, combined with Machine Learning algorithms and other classifiers, which mostly couldn't handle NaN values in data, so imputation was required. For building the best model, **CatboostClassifier** was used, which has many useful features and handles NaNs while also preventing overfitting. The final model with Catboost resulted in a high Multiclass AUC of 90% and accuracy of 65%. This model was used to build a **Streamlit** app, which allows for real-time implementation of the classifier and prints the top 5 genres for the given book with their respective probabilities. Additionally, the app has an Explore Books Data page, where additional information about the used dataset is presented.

One of the major areas for improvement is fixing the remaining genre class imbalance, as several genres still had 7-50 book examples, while others had more than 600. Although it is trained on few examples, it surely reduces the accuracy score, as given a new book from one of those genres, it won't be able to successfully predict it in most of the cases. Besides

that, only fiction genre books were scraped, as non-fiction genres are too many and available data in Armenian is scarce.

There were no available datasets for immediate usage from Kaggle, Github, or other sources, so all data was scraped. As the model's aim was to predict the genre of the book given details in Armenian, the dataset to use for the model also needed to be in Armenian. This introduced the problem of expanding the dataset with data in English, which required a translation, and several translation packages were chained together to reach promising results. This made the process of translation computationally costly and timely. If paid versions of automated translations, such as APIs, were used, the process would be much faster and more efficient.

To refine this project, expanding the dataset, adding also non-fictional genre books and obtaining better translation means may be helpful. Additionally, it would be beneficial to minimize the computational time cost of the model and fix the data imbalance in order to explore its opportunities to full scale.

REFERENCES

[1] Shiroya, P. (2021). Book Genre Categorization Using Machine Learning Algorithms (K-Nearest Neighbor, Support Vector Machine, and Logistic Regression) using Customized Dataset. International Journal of Computer Science and Mobile Computing, 10(3), 14–25. https://doi.org/10.47760/ijcsmc.2021.v10i03.002

[2] Gupta, S., Agarwal, M., & Jain, S. (2019). Automated Genre Classification of Books Using Machine Learning and Natural Language Processing. Confluence. https://doi.org/10.1109/confluence.2019.8776935

[3] Desai, P. (2021). Book genre prediction. International Journal for Research in Applied Science and Engineering Technology, 9(10), 593–599. https://doi.org/10.22214/ijraset.2021.38409

[4] Scraping a site — Web Scraper Documentation. https://webscraper.io/documentation/scraping-a-site

[5] Gensim: Word2vec Embeddings. https://radimrehurek.com/gensim/models/word2vec.html

[6] YerevaNN. (2015). GitHub - YerevaNN/word2vec-armenian-wiki: Testing word2vec on Armenian Wikipedia. GitHub. https://github.com/YerevaNN/word2vec-armenian-wiki.git

[7] GeeksforGeeks. (2024, April 29). CatBoost in Machine learning. https://www.geeksforgeeks.org/catboost-ml/

[8] GeeksforGeeks. (2023, January 19). Understanding TF-IDF (Term Frequency-Inverse Document Frequency). https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

[9] Streamlit. A faster way to build and share data apps. https://streamlit.io/

[10] Bookinist book, (2024). https://books.am/am/catalog/product/view/id/96806/category/2/.