CS6370: Natural Language Processing
Recommend books to Age groups

Team14
Anvesh B (CS14B037) || Satish G (CS14B042)

Indian Institute of Technology Madras

Abstract. The first step of the recommendation of books to children based on their age is to get a list of books with in their readability level then rank them based on scores given by the recommendation engine

Keywords: Readability · Recommendation · Children books .

1 Problem Statement

Our aim for this project is to develop an application which recommends books(Fiction, Short stories, Fairy Tales, History, Educational, Instructional, Picture books) to children. Recommendation of books is done considering various factors such as age of the user, readability level of a book (for generalized and personal recommendation), and books read by the user previously(for personal recommendation).

2 Methodology

2.1 Readability Classification

We built a machine learning model to classify the books based on their Readability factor. We used the Weebit Corpus to train, validate, and test our model. We extracted the traditional features(5), and the lexical features(14) of each example in the corpus. These 19 features are given as input to our model.

Corpora
The WeeBit Corpus (Vajjala & Meurers 2012)

- The WeeBit Corpus is a compiled from both **WeeklyReader** and **BBC-Bitesize**. It is a corpus of 3125 texts belonging to 5 readability levels, corresponding to the age(7-16) group of children.
- WeeklyReader is an educational newspaper, with articles targeted four grade levels (Level 2, Level 3, Level 4, and Senior), corresponding to children between ages 7–8, 8–9, 9–10, and 10–12 years.

- BBC-Bitesize is a website with articles belonging to four grade levels (KS1, KS2, KS3 and GCSE), corresponding to children between ages 5–7, 8–11, 11–14, and 14–16 years
- To cover a broad range of non-overlapping age groups, we used,

WeeBit class	**Age (years)**	**Reading level**
Level 2	7–8	1
Level 3	8–9	2
Level 4	9–10	3
KS3	11–14	4
GCSE	14–16	5

- To avoid classification bias towards a particular class because of biased training examples,
  - Training data included 450 documents from each level,
  - Validation data included 50 documents from each level,
  - Test data included 125 documents from each level.
  - In total, we trained our model with 2250 texts and tested it with 625 texts, spanned across five grade levels.

Preprocessing of Data
Before extracting the features from examples, we preprocessed the available data by following the below four steps:

- Few texts included words from other languages(e.g. German names) which are not in ASCII format. Hence, we **decoded the text from utf-8 format to ASCII**.
- We **removed all the punctuations**('.', ',', '?', '!' etc.) from text as they add noise to feature extraction.
- In the last step, we **tokenized the text** to use them during extraction.

Features
To indicate (1) lexical richness of a text, (2) difficulty in reading a text, (3) difficulty to understand a text, we extracted the following 14 features from each text.

- **Traditional Features**
  - **MLS:** It is the average length of a sentence in words.
  - **NumSyll:** It is average number of syllables in a word.
  - **NumChar:** It is the average number of characters in a word.
  - **Flesch-Kincaid score:**
    - It is calculated to indicate how difficult a passage is to understand. High score indicates the text is difficult to understand.
    - The formula was designed based on various experimentations performed.
    - FK score $= 0.39(\frac{TotalWords}{TotalSentences}) + 11.9(\frac{TotalSyllables}{TotalWords}) - 15.59$

- **Coleman-Liau readability formula:**
  - ∗ This formula is designed with an intention that the word length in letters is a better predictor of readability than word length in syllables.
  - ∗ CL formula $= 0.0588L - 0.296S - 15.8$
    where, L is average no. of letters per 100 words, and S is average no. of sentences per 100 words.

– **Lexical Features** In a text, the lexical words in it give meaning, and information about the text. In general, lexical words are simply nouns, adjectives, verbs, and adverbs. Other plane words like articles, pronouns, prepositions etc are called non-lexical words.

- **Lexical Density (LD)**
  - ∗ It is the ratio of number of lexical words to the total number of words.
  - ∗ LD $= \frac{No.oflexicalwords}{Totalno.ofwords}$
- **Type-Token Ratio (TTR)**
  - ∗ It is the ratio of number of word types(T) to total number of word tokens(N).
  - ∗ Many other alternative transformations of TTR came into existence.
  - ∗ TTR $= \frac{T}{N}$
  - ∗ Root TTR $= \frac{T}{\sqrt{N}}$
  - ∗ Corrected TTR $= \frac{T}{\sqrt{2N}}$
  - ∗ Bi-logarithmic TTR $= \frac{LogT}{LogN}$
  - ∗ Uber Index $= \frac{Log^2T}{Log(N/T)}$
- **Lexical Word Variation (LV)**
  - ∗ It gives the ratio of number of lexical word types($T_{lexical}$) to total number of lexical word tokens($N_{lexical}$).
  - ∗ LV $= \frac{T_{lexical}}{N_{lexical}}$
- **Noun Variation (NV):**
  - ∗ It gives the portion of nouns(N_{noun}) compared to all the lexical words($N_{lexical}$) in a text.
  - ∗ NV $= \frac{N_{noun}}{N_{lexical}}$
  - ∗
- **Adjective Variation (AdjV):**
  - ∗ It gives the portion of adjectives(N_{adj}) compared to all the lexical words($N_{lexical}$) in a text.
  - ∗ AdjV $= \frac{N_{adj}}{N_{lexical}}$
  - ∗
- **Adverb Variation (AdvV):**
  - ∗ It gives the portion of adverbs(N_{adv}) compared to all the lexical words($N_{lexical}$) in a text.
  - ∗ AdvV $= \frac{N_{adv}}{N_{lexical}}$

- **Verb Variation-2 (VV-2):**
  - ∗ It gives the portion of verbs(N_{verb}) compared to all the lexical words($N_{lexical}$) in a text.
  - ∗ VV-2 = $\frac{N_{verb}}{N_{lexical}}$
- **Verb Variation(VV)**
  - ∗ It gives the ratio of different verb types(T_{verb}) to total verb tokens(N_{verb}) in a text.
  - ∗ Alternate variations are also available. Namely,
  - ∗ Verb Variation-1 = $\frac{T_{verb}}{N_{verb}}$
  - ∗ Squared Verb Variation-1 = $\frac{T_{verb}^2}{N_{verb}}$
  - ∗ Corrected Verb Variation-1 = $\frac{T_{verb}}{\sqrt{2N_{verb}}}$

Model

– We trained our Multi-layer Perceptron with the parameters,
  - NUM_F : 19(5+14)
  - sizes : 10(1 layer)
  - lr : 1e-2
  - $l_2(\alpha)$: 0.01

– For the above model,
  - Train_acc : 60%
  - Valid_acc : 68%
  - Test_acc : 59.5%

2.2 Recommendation

Data

– **User data**
  - We built a personal recommendation model which recommends books to a user based on the user's age, books read by the user previously, and the user's ratings for the books.

– **Library data**
  - We compiled the library by crawling two websites which contain articles written for children:
    - ∗ **Time Time For Kids:** It's a section(www.timeforkids.com) of the TIME magazine, which publishes articles exclusively for children. We collected a sample of **471** articles from TIME.
    - ∗ **FirstNews ChildrensNews:** We crawled this website(live.firstnews.co.uk) that contains news articles written for children and collected **72** articiles

Models

- **Popularity based**
  - When a **new user** signs up, we don't have any information about their history of reading books. It is difficult to recommend books for them
  - Using this model, we get a list of books(articles) which are readable by the user based on the user's age and then prioritize the books based on their **popularity**(avg.rating of the book) among the existing users.

- **Content based algorithm**
  - This algorithm is used to recommend books for the existing users based on their previous read books.
  - Based on the users-books(articles) data i.e, which users read which books, we compute the representations of books by the below method,
    - Construct a matrix $M_{users \times books}$ consisting of only 0,1's where the element $M_{ij} = 1$, if i^{th} user read the j^{th} book and 0 otherwise.
    - Factorize the matrix into two matrices i.e, $M = U \times B$ where one matrix(U) is user representations and the other(B) is books representations. We used **Singular Value Decomposition(SVD)** for the factorization.
  - After computing the representations we recommend the books by computing the **cosine similarity** between the un-read books and the previous read books of the user.

3 Results

Feature Set	**# Features**	**Accuracy** (%)
Traditional Features	5	58.2
Lexical Features	14	44.3
Traditional+Lexical Features	19	59.5

4 Discussion and Future work

We explored different kinds of features which affect the readability factor of a story book. We extracted only the traditional features(5), and Lexical features(14) from texts. Syntactic features can also be extracted which will help the model in better learning of texts.

We used Multi-Layer Perceptron(MLP) model to classify the texts. We executed 300+ different experiments tuning the parameters(lr, num_neurons, α, num_feat) of the model. The bench-mark accuracy given for the model is 70%. We achieved 59.5% accuracy. All the features don't play equal role in defining readability factor. We can prioritize the features and consider only the important features using correlation algorithms.

We initially crawled books(classified by genres) from different websites. Surprisingly, more then 80% of the books have fallen under a single class. Hence, we switched to recommending articles where we got better results.

For building the library of articles, we crawled media websites with articles written for children. After crawling, we had to process the texts and strip unnecessary content using different methods. We got exposed to various crawling methods, and libraries available online for doing this.

We tried extracting features only for a part of the text. We ended up getting invalid feature values. Hence, we extracted features from the entire text instead of just from a part of text.

For future work in recommendation based system, we can build a library of books. With this work, we can recommend books to a user based on the genre of book too. We didn't have real-world based user's data while building personal recommendation model. We generated artificial data of **10,000** users' information(user's age, history of books read, ratings for the books read) to train the model. We used only content-based approach while building the model. This can be improved by also using collaborative based approach. Also, the user/books representations can be improved by using better methods for matrix factorization.

References

1. Sowmya Vajjala, Detmar Meurers, "On Improving the Accuracy of Readability Classification using Insights from Second Language Acquisition", http://aclweb.org/anthology/W12-2019
2. Sowmya Vajjala, and Detmar Meurers. On the applicability of readability models to web texts. http://www.aclweb.org/anthology/W13-2907
3. https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/