Make n' Model: A Model Kit Recommendation System

Erik Anderson **Farhan Ariff Bin Halis Azhan** **April Choe**
Carter Gracin **Andrew Sawicki**
University of Michigan – Ann Arbor
{farharif, erikrolf, aprilyc, cgracin, sawickid}@umich.edu

1 Introduction

The hobby of scale modeling (e.g. cars, planes, tanks, etc.) is popular across the world, amassing over 240,000 users on r/modelmakers–a subreddit for model making hobbyists, and gives homage to some of the most impressive innovations in human history. Each model contains intricate parts that require meticulous work to reach a finished product, in which hobbyists display or sell their work. Part of that work entails painting each piece individually with multiple different types of techniques. With over 300 different brands of paints, keeping track of which paints you own, and which models you can create is complicated and can limit a hobbyist in finding unique models to build.

2 Project Description

2.1 Current Problem

The main objective of our project, Make n' Model, is to recommend vehicle model kits (e.g. planes, tanks, cars, and boats) based on user inputs and preferences and minimize the overall waste of paints in model making. We aim to focus on the user's paint inventory as paints have to be bought separately and is thus the most restrictive aspect of the process. Another issue can be found in the assumed difficulty of certain models and how hobbyists can start models just because they have the paints, but don't intend to spend the time that is required to complete the model. Most models do not have an associated difficulty rating or list of the paints they use, meaning that model makers typically have to buy the model, look at the instructions and figure out if they have the required paint, and if the model falls within their skill level. This often leads to model makers wasting money on models they can't build, or having to go spend more money to buy the additional paints that the model requires.

2.2 Proposed Solution

Our main interface is an interactive website, where users can create an account to maintain a database to reflect their personal inventory. Paints added by the user are tracked in the database and organized by the brand of the paints. An option to mark paints as low is also available. Currently, our main database contains all the paint brands that the website has, providing autocomplete to a user's query request when adding paints. On each user's profile, paint usage status, collection insights and statistics, and navigational buttons to modify or view the user collection are shown.

The key aspect of the project is the recommendation system to present users with the most relevant results for their paint inventory and difficulty preference. Each model kit has a corresponding instructional PDF containing the paints needed for construction. A unique code distinguishes each paint by its brand and color and can be used to match the paints in the user's database. User ownership of the paints needed for the model kit, determined by the matching paint codes, contributes to our recommendation algorithm. Additionally, a difficulty score will be calculated for each model, which will contribute significantly to its overall rank score. A component of the difficulty score calculation will be based on the painting score, number of steps, keywords present, and word count per page in the instruction PDF. Key-words will be a predetermined list extracted from various PDFs that indicate difficulty. The score calculated for user ownership of the necessary kit paints and the score for model difficulty will be weighed respectively and multiplied to determine the ideal weighting scheme for computing the overall ranking score. This overall score will aid in the selection of recommended models for the user.

For our recommendation interface, the user can choose a difficulty level of easy, medium,

or hard which are predetermined based on thresholds of the difficulty score. We will use this threshold to recommend models that the user can build with the paints they currently own so no additional paints need to be bought. We also want to incorporate similarity scores into our program, so to do this we will be providing 10 additional recommendations to the user that contain the models that use the greatest number of paints the user has and are the difficulty rating that the user has chosen. These will be divided into categories based on how many paints the user is missing from that model's required paints. These categories are "Missing 1 Paint", "Missing 2 Paints", "Missing 3 Paints".

An important aspect of our program is gauging how well it works. To do this, we will create a "Golden Standard" dataset of 20 models of varying difficulty. We will manually go through these model's instructions and calculate the key aspects that contribute to our recommendation and difficulty score and create our manual recommendation and difficulty score for these 20 models. We will also use the golden standard to create thresholds for the difficulty scores to create the different categories. We will then run our algorithm on these "Golden Standard" documents and assess the similarity between the program ratings and our manual ratings.

3 Related Work

Text difficulty classification is a relevant topic in natural language processing, aiming to determine the complexity of a text. Curto et al. (2015) applied this text difficulty classification to classify educational material utilizing a combination of feature extraction and natural language processing tools. Relevant features (i.e. number of words, parts of speech, etc.) were extracted from the texts and weighted appropriately to determine the text readability score.

Multi-topic classification Rahman and Akter (2019) compared the usage of three classifiers–Decision Trees, K-Nearest Neighbors, and the Multinomial Naive Bayes–for multi-topic classification. For preprocessing the texts, they experimented with three techniques to determine that stop word removal and stemming improved accuracy of the classifiers, while lemmatization had no impact on performance. Each model's results were, then, evaluated using the accuracy, precision, recall, and f-scores, where the overall accuracy of the Multinomial Naive Bayes (91.8%) was found to be comparatively higher than K-NN (82.6%) and Decision Trees (79.6%).

Additionally, McCallum and Nigam (1998) compared the usage of the Multinomial Naive Bayes and multivariate Bernoulli models. They concluded that the Multinomial Naive Bayes performs uniformly better and performs better for documents with varying length while the multivariate Bernoulli model reduces error for real-life corpora by approximately 27%. Given that our dataset contains a large range for document length and the model's consistent high performance, we chose Multinomial Naive Bayes to combine with the feature extraction, as conducted by Curto et al., for our text difficulty classification.

Pitigala et al. (2011) explored a personalized retrieval system using text classification by comparing the performance between Naive Bayes and Support Vector Machines (SVM) trained on a smaller set of texts when classifying full texts. They concluded that, given the consistent higher performance of Naive Bayes compared to SVMs, Naive Bayes would be a more effective choice in building a personalized retrieval system.

Lastly, similar to our project, Supercook is an application that allows users to search for recipes that utilize ingredients they already have in their home. The app functions as a reverse search engine that retrieves recipes from an existing database to present numerous recipes that utilize a combination of ingredients selected by the user. With more than 1 million downloads for the mobile application, Supercook provides a good reference for our application to understand effective interface and algorithmic decisions. Similar applications also exist for other industries, including a fashion application, called Cladwell, that allows users to store their clothing pieces and displays daily outfit suggestions based on their clothing inventory. Exploring these applications and their implementations can aid in our development by revealing components to consider in our retrieval system that we had not considered.

4 Datasets

4.1 Data Collection

The main repository where we are collecting our data comes from a website called scalemates.com. This website contains repositories on different companies' models, their instructions, and all of the

paints that the company offers. We wrote a scraper for this website using beautifulSoup4 to retrieve all of the paints, their hex color code, the name of the paint, the paint type (acrylic, lacquer, oil), the shine type (Matte, Gloss, etc..) and added everything into our database, for every brand that the website offers. In our database, we associate each new brand we add into our database with a unique brand identifier. This identifier is automatically incremented and decremented when a brand is added and deleted and is used to annotate this brand to quickly access it later. We do a similar thing with each paint that is added to our database. We use a unique paint identifier that is automatically incremented and decremented to annotate each new paint and quickly access it later. With this data, we are able to autocomplete paint color and brand name queries when the user is adding a paint color. Through this method, we are also able to retrieve the instructions for each model that is on the site. The instruction PDFs retrieved from the website were run through Google's Document AI to extract the text from the PDFs. Document AI was utilized as many of the PDFs contained PDF images and the AI was trained for these types of text extraction. We are also taking into account the length of the PDF. For each PDF, we will be calculating a score which takes into account many different aspects from the classification algorithm, and assigning a unique instruction identifier to each document which associates the score and related data to that document identifier, which automatically increases and decreases with the addition or deletion of the doc.

4.2 Data Annotations

In order to train our difficulty classifier, we first had to come up with labels that we intended the model to learn to predict. We decided on using three labels: 'easy', 'medium', and 'hard.' This is a way to convey to the user an intuitive sense of difficulty without adding unnecessary complexity to our learning model. Our labeling process consisted of picking instruction PDFs at random and sorting them into our three difficulty categories based on a list of factors, including but not limited to: number of steps, scale of model, size of parts, and methods of construction. We continued until we had 25 labeled documents in each category.

5 Method

5.1 Preprocessing

Instruction PDFs were processed through the Document AI and outputted as JSONs that contain the extracted text from the PDF. First, the paints used in the model and the parts required for the model were extracted from the text. Since paint and part names are not indicative of difficulty, we removed them from the text before pipelining the output to the language classifier. The paints and quantity of parts are instead stored separately for usage in future calculations. Often, the instruction PDFs contained multiple translations for the text to cater to a global audience. As a result, we chose to utilize FastText-LangDetect, a Python library, to remove any language other than English to reduce computation time. Lastly, we removed stop words, performed stemming, and tokenized the text to pipeline into the difficulty classifier.

5.2 Difficulty Classification

Given a table of paints and their brand name, paint name, paint code, shine, and type, we took a user's input or collection of paints to search for models that use exact or similar paints. A user can have any number of paints, as long as they belong to one of the supported brands from our database. When we parse the model instruction data, we will be assigning a list of paints to each, and this is what we are searching for with the highest importance. The user will not have to include their paints with each search, but instead we will keep track of each user's paint inventory as paints are added or removed. When retrieving models, we will find the intersection of paints in the user's inventory with the list of paints associated with each set of instructions and take into account the number of paints that the user does not have. This will factor into how relevant each model is in our search, and models will be classified as ready to build, or missing 1, 2, 3 or more paints.

There are three aspects of the instructions that we used to evaluate the difficulty of the models, which are paint score, parts per page multiplied by scale score, and Naive Bayes score.

5.2.1 Paint Score

The first aspect we consider in calculating difficulty score is paint score. Paint score is a metric we establish by dividing the number of unique paints referenced in the instructions by the

number of parts. This ratio grows, either as the number of unique paints increases, or if different paints are used on the same part. Conversely, the score will decrease if the number of unique paints decreases, or if the same paint is used on multiple parts. For example, if a model needs 5 colors of paint, but has 10 parts (or any ratio less than 1), we can assume that the same paint is used on several parts, decreasing the difficulty of the painting. Alternatively, a ratio greater than one indicates that certain parts require more than one paint, increasing the difficulty.

5.2.2 Parts Per Page Score, Scale Score

The second aspect we consider in calculating difficulty score is parts per page score multiplied by scale score. Scale score means that a model's difficulty can be assessed by its scale. We took into consideration the most common model types of each scale and assigned a rating to each scale, ranging from 0.1 to 1. Any previously-not-encountered scales that were found during data scraping were assigned the rating of the closest scale in our references. The parts per page score, or number of parts divided by the number of pages, was included in our formula to assess how complex the instructions were, as well as how large the parts were. We believed the interaction between these two terms was a good indicator of difficulty, as a larger number of small parts should, in theory, indicate a higher difficulty than a smaller number of large parts.

5.2.3 Naïve Bayes (NB) Score

The last aspect we consider in calculating the difficulty score is Naive Bayes score. Using our Naive Bayes classification model, we received a difficulty label for each model. This label was converted into a decimal, turning 'easy', 'medium', and 'hard', into 0.3, 0.6, and 0.9, respectively.

5.2.4 Weighting

To get a more accurate score, a weighting scheme was included in our formula. Each of the above three scores received a weight, based on how important we deemed the terms in the final classification. The weights we settled on were 0.3 for paint score, 0.3 for parts per page score * scale score, and 0.4 for Naive Bayes score.

5.3 Normalizing Data

To reduce redundancy, we normalized our difficulty scores by dividing the difference of each score by the range of difficulty scores. This gave us scores between 0 and 1, indicating lowest difficulty and highest difficulty, respectively. To create thresholds for our final difficulty classifications, we used our formula to calculate the difficulty scores of the 75 previously, manually labeled documents. We took the average scores for each category and found the median of the easy/medium and medium/hard averages to set our maximum 'easy' score and maximum 'medium' score.

6 Results

We are evaluating the Naive Bayes multinomial algorithm on F1 score and overall accuracy. The Naive Bayes algorithm was evaluated using 75 documents within our golden standard using Leave-One-Out Cross Validation.

Class	Precision	Recall	F1
Easy	0.48	0.44	0.46
Medium	0.52	0.45	0.48
Hard	0.32	0.42	0.36

Table 1: Evaluation scores from N.B. alg.

The results are shown in Table 1 and represent how well our algorithm can classify test data based on the words in the document.

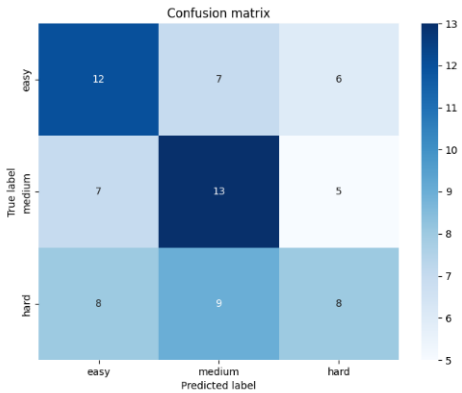

Figure 1: Confusion Matrix on N.B. alg.

From the results shown in Figure 1, we can see that our algorithm classifies medium difficulty instructions with the most confidence, followed by easy, then hard.

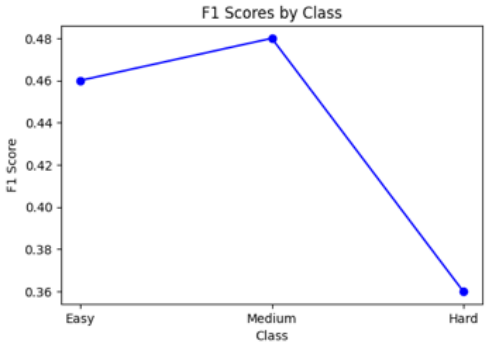

Figure 2: Classification data.

The hard classification performing the worst within the F1 score comparison (Figure 2) could be attributed to the ambiguity between medium and hard classifications within the golden standard. On the other hand, easy and medium classifications had very similar relatively high F1 scores which could be attributed to the clear distinction between the differing types of instructions.

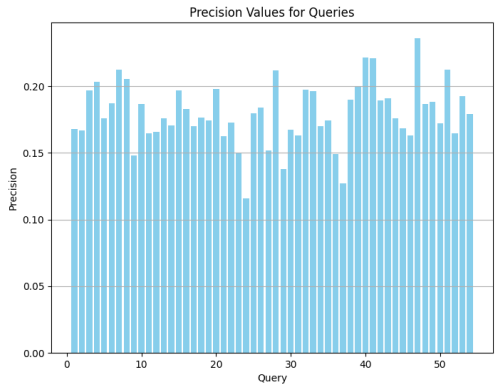

Figure 3: Precision values for queries

For the evaluation of our ranked retrieval system, we are using mean average precision (MAP) and overall accuracy. The MAP of the system was ~0.18. We found from the MAP; the system did not perform with high precision after tests of 5554 queries to the system (Figure 3). We can attribute these low numbers and low overall accuracy to the fact that we were not able to get all the paint colors off of the PDFs that we scanned, and our database did not reflect the true paints needed to complete the model.

7 Conclusion

The information retrieval system that we coded serves its purpose. We aim for the program to return to the user the different models that the user can build depending on the paint that the user has. The user will get an output, based on the paint that they have, models that they can build with all the paint they have, models that are missing one paint, models that are missing two paints, three paints, and models that are missing more than three paints. In addition, the returned models' will be annotated with colors; green for easy, yellow for medium, and red for hard to indicate their difficulty.

We deemed the system successful in recommending to the users the models that they can build with what paint they have; thus they do not need to keep track of the paints they have or waste money and resources to buy new paints.

7.1 What Did Not Work

In the initial work of this project, we consider other indicators to be included in calculating the models' difficulty score, which are keywords score and image analysis score. Attempting to analyze the difficulty of different images within the PDF and extract useful information from the analysis did not work as we were unable to find an efficient way to implement this. We also found that we were unable to analyze all models among all categories. In order to run our PDFs through Google Document AI, we needed to download each PDF, and none of our computers had enough space to hold this data. We therefore limited our data to only Tamiya models, and still had over 20GB+ of PDFs.

7.1.1 Keywords Score

Initially, we consider keywords score. At first, we thought that we could use keywords to indicate the models' difficulty score. For example, verbs like cut, put, glue or nouns like the tools, such as scissors, glue, or brush, can be used to indicate the difference in difficulty score between models.

7.1.2 Image Analysis

After that, we consider doing image analysis on the image from the PDF. The instruction PDFs have images in them, which shows how many parts are there in the models. We want to use image analysis to extract the number of lines in these models. We assume that the more lines there are in the PDF the harder it would be. From there, we can use this number of lines to calculate the difficulty score. Unfortunately, we did not do image analysis as the program failed to differentiate between the images and the text, which caused errors.

7.2 Considerations for Future Works

7.2.1 Feedback Option

One of the limitations that occurred during our work is the difficulty classifier. We are only able to personally classify 75 instruction PDFs into easy, medium, and hard, 25 files for each difficulty. After that, we use Naive Bayes program to pre-classify the remaining instruction PDFs into the three difficulty categories. Interestingly, our Naive Bayes classifier has accuracy of 44.44%, granted that the class probability for each difficulty is 33.33%.

To overcome this limitation, one of the ideas that can be implemented is to include a feedback option for each model that user has attempted to build. In our opinion, this feedback option will be more helpful and accurate in classifying the models into its difficulty. We think so because this feedback will come from people that have personally experienced the difficulty in building the model and will be more accurate than using a program to classify it.

Expectedly, this feedback option may not be effective since users might not feel inclined to evaluate the models they build. Therefore, despite appearing to be a better gauge of difficulty, it was anticipated that there wouldn't be much user participation in this option.

For now, difficulty classifiers using Naive Bayes serves as a means for us to predetermine the difficulty score for each model, while the feedback option may serve as a way to adjust the predetermined difficulty to its actual difficulty.

7.2.2 Include More Models

Another limitation in our work will be limited model availability. The models that we have in our database only come from scalemates.com. Furthermore, the models that we consider are only vehicles. Admittedly, this means that the models that can be returned to the users will be limited to vehicles that come from scalemates.com.

With this in mind, our consideration for future work will be to include more models' categories, from other websites. We hope that by including more model availability in our database, it can increase user satisfaction when they use our information retrieval system.

7.2.3 Try Other Classification Methods

Another limitation in our work will be that we only use Naive Bayes in difficulty classification. To improve this, we will want to use other classification methods, and if they have better accuracy than Naive Bayes implementation, to include it in our implementation.

For example, neural networks and bootstrapping are both valid text classification methods that could have been used to create difficulty rankings. Bootstrapping would be a good option, due to the lack of confidently labeled documents. That way, we could grow our training set as we run the classifier, most likely increasing accuracy.

7.2.4 Improve Training Process

As said above, we only personally classified 75 models into its difficulty, train on these 75 files and then use it in classifying the remaining models' difficulty.

One way to improve this limitation is to classify more data and validate them first before using it to classify the other models, or split the classified data into training and validation files.

7.2.5 Difference in Human and Machine Classification

When we classified these models into its difficulty, we humans personally see the models' PDF and can visually and internally assess if the model is difficult or not by its text and pictures from multiple reasons like scale, number of parts, the number of paints needed, and number of pages. On the other hand, the input data that we give to the Naive Bayes for our difficulty classification is only text from processed-PDF-to-JSON, and the one that we have filtered to be English text only.

One way to improve our difficulty classification is to include images that have been processed into the difficulty classification program, so the program can more similarly "assess" the models in the same way that we humans do.

7.2.6 Translating the Paints Between Brands

When searching for paints, the only brand we used was Tamiya. There are several brands of paints on the market, and many of these paints are essentially the same color as Tamiya paints. We had planned to allow these other brands to be added to the user's inventory, and we collected data to facilitate conversions between brands, but we were not able to finish this feature in time. In the future this would be a helpful option to make our application accessible to a wider range of users.

8 Individual Contributions

Erik Anderson contributed to the development of the Naive Bayes multinomial algorithm, overall pipeline of calculations of the difficulty score, frontend design on the website, and using Google Cloud Document AI API.

Carter Gracin contributed to the development of the front and backend of the website. This includes the HTML and Flask/Python routing for the frontend, the development and design of the SQL databases and the SQL queries that returned the paints that matched the user's inventory. He also contributed to the development of the Naive Bayes algorithm and the design of our difficulty classification algorithm.

April Choe contributed to the development of the Naive Bayes algorithm, as well as the overall difficulty score algorithm and the integration of the Document AI. She helped with the development, linkage, and pipelining of all the separate parts of our algorithm through one main function. This allowed us to effectively perform all the functions and ensure that data was efficiently transferred between each function.

Andrew Sawicki worked on the difficulty score classifications, writing the formula, scraping data for model scales, and writing scripts to populate the database. He also worked on pipelining data through our classification and retrieval systems and aided in the formulation of the Naive Bayes algorithm.

Farhan Ariff Bin Halis Azhan worked on the Naive Bayes algorithm, classifying documents and creating output files. He also worked on front-end development.

References

A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," 1998.

"Cladwell | Simplify Your Life With A Capsule Wardrobe." Cladwell, https://cladwell.com. Accessed 19 Mar. 2024.

M. A. Rahman and Y. A. Akter, "Topic Classification from Text Using Decision Tree, K-NN and Multinomial Naïve Bayes," 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 2019, pp. 1-4, doi: 10.1109/ICASERT.2019.8934502.

P. Curto, N. Mamede, and J. Baptista, "Automatic Text Difficulty Classifier," in Proceedings of the 7th International Conference on Computer Supported Education., 2015.

S. Pitigala, C. Li and S. Seo, "A comparative study of text classification approaches for personalized retrieval in PubMed," 2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), Atlanta, GA, USA, 2011, pp. 919-921, doi: 10.1109/BIBMW.2011.6112503.

Supercook: Recipe Search by Ingredients You Have at Home. https://www.supercook.com/#/desktop. Accessed 19 Mar. 2024.