Enhancing Classification Performance with Combined Embedding Techniques and Machine Learning Models

Cuong Pham & Trang Doan

Abstract

When searching for products on Amazon, it is common to encounter items with similar names that are entirely different from what you are looking for. This confusion arises from the platform's vast and diverse product catalog, where slight variations in product titles can lead to significant differences in the actual items. This issue highlights the importance of precise search algorithms and effective categorization to enhance the user experience. The goal of this paper is to address one aspect of this challenge, starting with better classification of products.

1 Introduction

Amazon features an extensive inventory across a wide range of categories, making efficient navigation and product discovery critical for customer satisfaction and business success. Accurately grouping products can therefore improve engagement metrics and enhance user's experience.

Among the 30 product categories featured, three particularly noteworthy ones are *Home & Kitchen*, *Books*, and *Beauty & Personal Care*. Each category is marked by a large assortment of products, but an interesting challenge arises from the similarities in product names across these diverse categories. For example, a "blender" could refer to a kitchen appliance in *Home & Kitchen*, a technique or tool in a beauty makeup tutorial under *Beauty & Personal Care*, or be the title of a book discussing social interactions in the Books category. This overlap in product names can lead to customer confusion, inadvertently guiding them to the wrong category, and consequently affecting their shopping experience. The significant volume of reviews in these popular categories further underscores the urgency for sophisticated, precise product classification systems. Such systems are vital not only for enhancing user experience by ensuring customers find exactly what they are searching for but also for boosting operational efficiency by accurately sorting and managing inventory.

The aim of this project is to develop a robust machine learning model capable of classifying products into these three specific categories using features such as: product title and product features. We will implement several modeling techniques, from simpler models like logistic regression to more complex frameworks like neural networks, or even a hybrid approach combining multiple models. This exploration will help us understand the business challenge for accurate labeling as well as evaluating the advantages and disadvantages of each modeling technique in addressing this problem.

2 Dataset

The dataset used for this project is a subset of data from the 2023 Amazon reviews datasets, compiled by McAuley Lab. Due to the limitation of our computing capacity , we will randomly select 1 millions sample from each of the three following categories: Home & Kitchen, Books, and Beauty & Personal Care. A full description and guidance on how to download the data and read the file and be found here.

Within the scope of this analysis, we will chose **Main Category** as our response variable and **Title** and **Features** as our 2 predicting variables [2]. Our initial analysis started with the entire dataset for all models. However, during the modeling process, we noticed recurring misclassifications. This led us to believe that training with the entire dataset hindered the model's ability to learn from its errors, resulting in saturated predictions.

Therefore, we decided to create a subset focusing on recurring errors for further modeling and deeper analysis.

3 Pre-processing and Word Embedding Process

The LabelEncoder from sklearn.preprocessing is used to transform categorical labels into numerical format. This encoder assigns a distinct integer to each category, making it ideal for converting target variables or ordinal features. Given the categorical variable with values *Book*, *Home*, and *Personal Care*, LabelEncoder will map them to 0, 1, and 2, respectively.

Next, we need to convert words into numerical representations that are both meaningful and computationally efficient. We will use two methods: TF-IDF and BERT. To determine the best method for embedding, we fit a simple logistic regression model with 5-fold cross-validation for each embedding and compare them using classification error within each category.

3.1 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents. It works by first tokenizing the text into individual words, then calculating the term frequency (TF) which measures how often a word appears in a document. It also calculates the inverse document frequency (IDF), which assesses the importance of a word by considering how common or rare it is across all documents in the corpus. The TF-IDF score is computed by multiplying the TF and IDF values, resulting in a weighted representation of each word. This transformation converts the text data into a numerical format that reflects the significance of words, making it suitable for various machine learning applications such as information retrieval and text classification.

TF-IDF typically is a fast process as it often splits text into token (often words) then compute term frequency (TF) and inverse document frequency (IDF) for each token. It is a straightforward mathematical operations without deep learning or complex models, thus, often is a preferable method when speed is priority and semantic context of the text is less critical.[4]

Prior to fitting the TF-IDF method, we remove number and common words by using Scikit-Learn's TfidfVectorizer's built-in list of English stop words. We also lemmatize words to bring its back to its root or reduced form. The title matrix has dimension of 624602 and the feature matrix has dimension of 735850.

For TF-IDF presentation, the model displayed a consistent high precision and recall ratio for the home category while the others show mixed result.

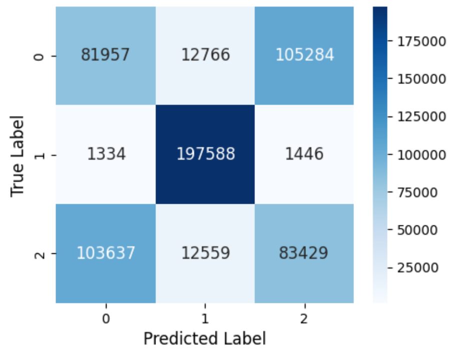

Fig. 1. Predicting Variable: Title. Embedding : TF-IDF

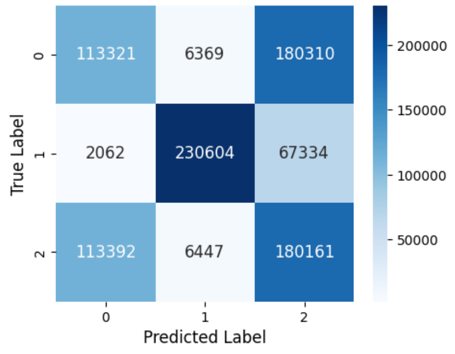

Fig. 2. Predicting Variable: Feature. Embedding : TF-IDF

Table 1. Performance Metrics for Logistic Model using Title variable and TF-IDF

Category	Precision	Recall	F-1 Score
Book	0.44	0.41	0.42
Home	0.89	0.99	0.93
Personal Care	0.44	0.42	0.43

Table 2. Performance Metrics for Logistic Model using Feature variable and TF-IDF

Category	Precision	Recall	F-1 Score
Book	0.5	0.38	0.43
Home	0.95	0.77	0.85
Personal Care	0.42	0.6	0.5

Looking deeper into the classification matrix above, we can see that the model often misclassified between the book and personal care category for each other while home category is likely miss-classified as personal care but not book. Later we will use these performances to compare with BERT embedding.

3.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) is an advanced NLP model that understands context by reading text bidirectionally, unlike TF-IDF, which relies on simple frequency-based statistics to represent the importance of words in documents without considering their contextual relationships. It was trained on large corpus data, including BooksCorpus (800M words) and Wikipedia (2,500M words). Prior to BERT, models like OpenAI's GPT used a unidirectional encoder, which processes text in a left-to-right or forward direction only. This bidirectional enhancement allows BERT to understand the context of a word based on both its preceding and following words, leading to better comprehension and more accurate predictions.

Using similar logistic model with cross validation, we can compute the miss-classification per category as shown in Figure 3 and 4.

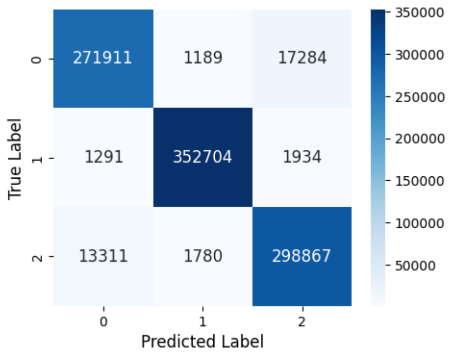

Fig. 3. Predicting Variable: Feature. Embedding : BERT

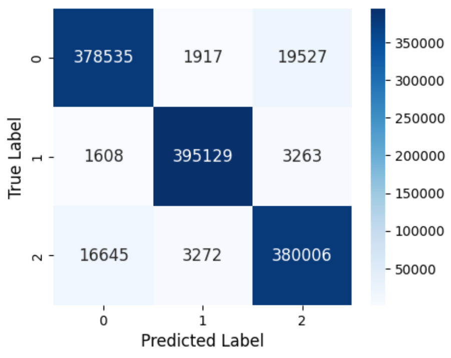

Fig. 4. Predicting Variable: Title. Embedding : BERT

By incorporating the context, BERT embedding clearly improve the separation between classes as well as miss classification rate. To visualize this, we randomly sampling

Table 3. Performance Metrics for Logistic Model with Feature variable using Bert

Category	Precision	Recall	F-1 Score
Book	0.95	0.94	0.94
Home	0.99	0.99	0.99
Personal Care	0.94	0.95	0.95

Table 4. Performance Metrics for Logistic Model with Title variable using Bert

Category	Precision	Recall	F-1 Score
Book	0.95	0.95	0.95
Home	0.99	0.99	0.99
Personal Care	0.94	0.95	0.95

1000 data points and applying PCA with 50 components. This allows the model to capture the most variances and remove noise from the data before applying t-SNE, a method that is extremely computation expensive when the data is in high dimensional space. Below are the scatter plots of 3 groups based on single variable predictor.

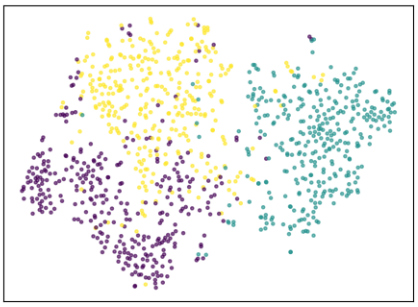

Fig. 5. Data Visualization With Title. Embedding : BERT

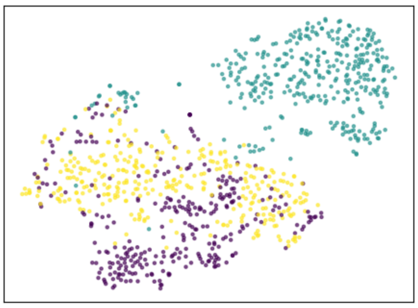

Fig. 6. Data Visualization With Feature. Embedding : BERT

As shown in Figure 5 and 6, the blue group representing, *Home*, is clearly separated from the others. Using title as a predicting variable seems to separate *Book* and *Personal Care* better than using feature.

Even though the above two graphs demonstrate a reasonable separation between groups and the combination of linear (PCA) and non-linear (t-SNE) data reduction techniques was utilized to capture the data pattern, it is important to remember that PCA assumes high variance is indicative of importance. This assumption is not always valid for BERT embeddings, where high variance does not necessarily correspond to the most semantically meaningful features. Consequently, PCA might emphasize components that are not essential for capturing the true essence of the text. Therefore, any interpretation of this plot needs to be approached with caution.

3.3 Data Size Reduction

Training the full dataset demands substantial computational GPU power for neural network and longer time for traditional machine learning algorithm on CPU. To address this, we selectively reduce the data based on the results of previous models, ensuring each category has an equal number of data points. We randomly choose 7,500 data points that the previous models predicted accurately and 5,000 data points that were misclassified. This approach helps to reduce the density of correctly predicted data, preventing overfitting and challenging the models with data they previously could not predict.

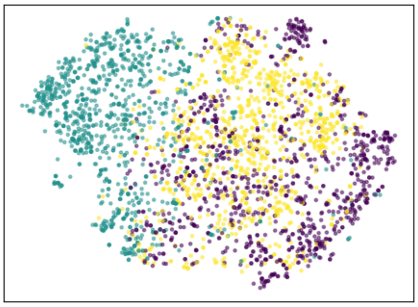

Fig. 7. Reduced Data Visualization

The training data, shown in Figure 7, is highly intermixed, particularly within the yellow and purple clusters. Although this is just a 2D representation, it offers insight into the data distribution and indicates where the data points cluster in the vector space.

4 Modeling and Results

We will compare models using the average accuracy scores and misclassification rates within each group. Since we anticipate new products being fed into the model daily, the runtime and computational resources required for generating predictions are also crucial factors in our final decision.

4.1 Logistic Regression Models

Previously, we implemented logistic regression models with single predicting variable and used their performance to compare and select our embedding technique. As shown in the scatter plot (Figure 5 and 6), the yellow and purple points are often intermixed, particularly when the feature is the only predicting variable. This aligns with the results of the confusion matrix provided above.

The logistic model often performs poorly when the probabilities of a data point belonging to multi classes result in tight probabilities. With large number of data, we can introduce bias to a certain categories, therefore we reduce the dataset to help the algorithm generalize better. Once we reduced the data size, we ran another logistic model with both predicting variables and fine-tuned two parameters: penalty and C-value for regularization. Based on our randomized search with 5-fold cross-validation, the optimal parameters were found to be $C = 0.2$ and L2 penalty. Table 5 and Figure 8 shows the result of logistic regression on a smaller dataset.

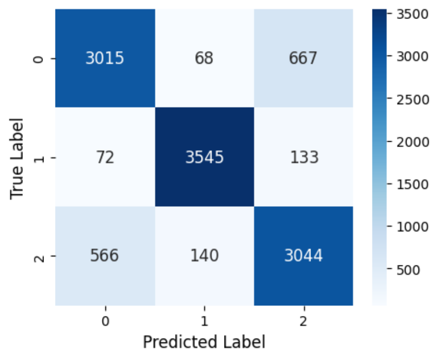

Fig. 8. Predicting Variable: Title & Feature. Embedding : BERT

Table 5. Performance Metrics for Logistic Model with Two Predicting Variables and BERT

Category	Precision	Recall	F-1 Score
Book	0.83	0.80	0.81
Home	0.94	0.94	0.94
Personal Care	0.79	0.81	0.8

4.2 K-nearest neighbor Models

From our visualization of 1000 samples, we can see how separable the 3 categories would be when using either feature or title as a predictor. This clear separation will be unlikely if we use a statistical embedding technique like TF-IDF since semantic meaning and context of words aren't incorporated in the numerical representation.

Using $k = 5$ and cross validation $n = 10$, we ran a KNN model that use cosine as the metric for distance and adjusted weight towards data points that are closer in term of distance. The cosine distance between two vectors $\mathbf{A}$ and $\mathbf{B}$ is given by:

$$\text{cosine distance} = 1 - \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

where $\mathbf{A} \cdot \mathbf{B}$ represents the dot product of vectors $\mathbf{A}$ and $\mathbf{B}$, and $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ are the magnitudes (or norms) of $\mathbf{A}$ and $\mathbf{B}$ respectively.

This metric is particularly useful when the vectors are sparse or when you are more interested in the orientation rather than the magnitude of the vectors. In this case, we are more interested in the relative values of the features rather than the absolute value, therefore, using cosine is more appropriate than regular L2 or L1 distance.

Based on the cross-validation (Figure 9), we can see that the model achieves very stable performance, closely resembling the logistic models mentioned above. However, the more interesting aspect is the significantly improved classification between Book and Personal Care. We do not expect this level of performance to hold if the dataset size were significantly smaller.

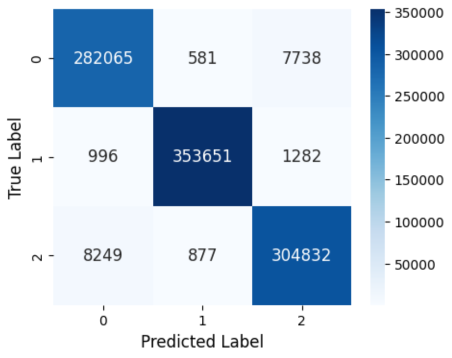

Fig. 9. KNN Model with $k = 5$

Table 6. KNN Cross Validation Score with $k = 5$

Performance Metrics				
0.9775	0.9774	0.9776	0.977	0.9772

Typically, in high-dimensional spaces, the distance between data points becomes less meaningful. As the number of dimensions increases, all points tend to become equidistant from each other. This often makes it difficult for KNN to distinguish between the nearest and farthest points effectively. Indeed, high-dimensional spaces are sparse, meaning that the data points are spread out and there are large empty regions. This sparsity can make it challenging for KNN to find meaningful neighbors.

However, some of these challenges can be mitigated when extremely large data set is used for training.

1. Dense Sampling: With a very large dataset, even in high-dimensional space, the data points are more densely packed. This dense sampling helps in better estimating the local neighborhoods and ensures that the nearest neighbors are more representative of the local structure of the data.
2. Redundancy: Large datasets often contain redundant information, which can help in stabilizing the KNN performance. Redundant data points can ensure that small variations or noise in the data do not disproportionately affect the model's performance.
3. Local Uniformity: With an oversized dataset, the local neighborhoods around each point can become more uniform, making it easier for KNN to find meaningful and relevant neighbors. This local uniformity helps in reducing the impact of the curse of dimensionality.[3]

We also created the KNN model using the reduced dataset and performed 5-fold cross-validation for each k value in the range between 5 and 1000. The cross-validation score didn't show noticeable changes until the k value was above 1200. We found that the smaller the k, the better the performance achieved.

However, KNN has a critical disadvantage in computing time. During training, it makes minimal effort by memorizing all training points, resulting in a testing time complexity of

$$O(nd + n \log k)$$

per each data point, where d is the number of dimension and n is the number of training data points and k being the number of neighbors. This can be significantly slow if the number of training and testing data points becomes arbitrarily large.

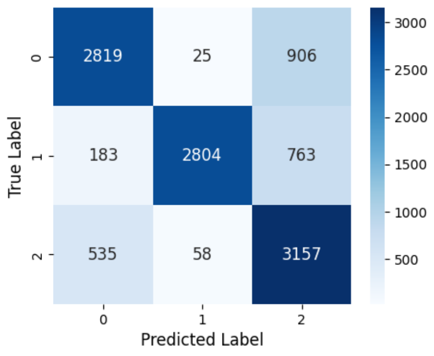

Fig. 10. KNN Model with $k = 1200$

Table 7. KNN Cross Validation Score with $k = 1200$

Performance Metrics				
0.78341	0.78244	0.79007	0.7864	0.77516

Table 8. Performance Metrics for KNN Model $k = 1200$ with 2 variables using Bert

Category	Precision	Recall	F-1 Score
Book	0.80	0.75	0.77
Home	0.97	0.75	0.84
Personal Care	0.65	0.84	0.74

4.3 Neural Network Model

As the number of data increases, traditional machine learning algorithm accuracy will eventually saturate at a certain accuracy level. However, deep learning model includes of multiple different layers which act finite as non-linear transformation can attain higher accuracy level using large number of data. Neural network models can growth big using a large number of layers to capture variability in data.

4.3.1 Model Architecture

The neural network model utilizes various layer types, such as embedding, dense, dropout, and LSTM (long short-term memory). Embedding layers are part of BERT's preprocessing step. Dense and LSTM layers perform non-linear transformations, while the dropout layer is applied after these transformations to prevent overfitting. Two types of neural nets are used:

- **One-path Model**: The model's inputs are combination of title and feature vectors by stacking those vectors together. Output is the predicted classification.

  $$\text{Input} \longrightarrow \text{Dense} \longrightarrow \text{LSTM} \longrightarrow \text{Dense} \longrightarrow \text{Output}$$

  Model configuration:

```
OnePathModel(
   (dense1): Linear(in_features=1536,
                    out_features=128)
   (dropout): Dropout(p=0.3)
   (lstm): LSTM(128, 128, batch_first=True)
   (dense2): Linear(in_features=128,
                    out_features=3)
 )
```

- **Two-path Model**: Rather than stacking the feature and title vectors, we train a hybrid model with these two inputs on separate networks. This approach maintains the original vector dimensions while extracting meaningful representations from both input sources. The combined features from those learned vectors can be more informative for the final prediction [1].

  $$\left.\begin{array}{ll}\text{Title} & \longrightarrow \text{Dense} \times (2) \\ \text{Feature} & \longrightarrow \text{Dense} \times (2)\end{array}\right\} \longrightarrow \text{LSTM} \longrightarrow \text{Dense} \longrightarrow \text{Output}$$

  Model configuration:

```
TwoPathModel(
   (path1_fc1): Linear(in_features=768,
                       out_features=128)
   (path1_dropout1): Dropout(p=0.3)
   (path1_fc2): Linear(in_features=128,
                       out_features=64)
   (path1_dropout2): Dropout(p=0.3)
   (path2_fc1): Linear(in_features=768,
                       out_features=128)
   (path2_dropout1): Dropout(p=0.3)
   (path2_fc2): Linear(in_features=128,
                       out_features=64)
   (path2_dropout2): Dropout(p=0.3)
   (lstm): LSTM(128, 64, batch_first=True)
   (final_fc): Linear(in_features=64,
                      out_features=3)
 )
```

4.3.2 Parameter Selection

Because neural network models can be computationally expensive in production, the main objective is to develop smaller models without sacrificing accuracy. Each layer the inputs pass through transforms higher dimensions to lower dimensions while preserving the semantic meaning of the original embedded inputs.

The original dimension is either 1536 for the combination vectors or 768 for the original title and feature dimensions. We tested and evaluated the following hidden layer dimensions: 512, 384, 256, 128, and 64. The initial dense layers gradually reduce the original input dimensions while retaining sufficient information for the subsequent layers.

Higher numbers of nodes in each layer require more computational resources. After several compression stages, the first dense layers in the `OnePathModel` and `TwoPathModel` have the lowest number of nodes at 128 and 64, respectively. Consequently, the `OnePathModel` reduces an input dimension of 1536 to 128, and the `TwoPathModel` reduces an input dimension of 768 to 64.

4.4 Result

Both models were trained on 80% of the reduced dataset and tested on the remaining 20%. Using a higher number of epochs can lead to overfitting on this relatively small dataset, so training was stopped at 40 epochs. As shown in Figure 11, the loss for both models fluctuates around 0.2 after 30 epochs, with the two-path model demonstrating greater stability and less variation in loss values after 35 epochs.

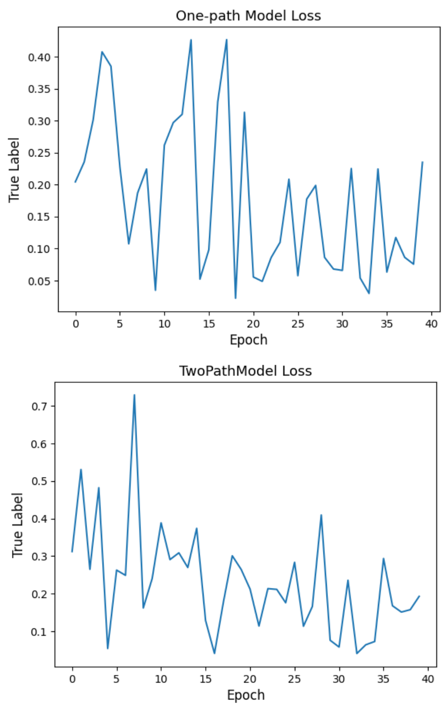

Fig. 11. Loss Value of Each Epoch

The following Figure 12 shows the confusion matrix result of both neural network models on the test set. The overall accuracy and f1-score of the one-path model are 87.63% and 87.60%, respectively. Those values of the two-path model are 87.83% and 87.79%. The detailed results of both models are shown in Table 9 and 10.

Table 9. Performance Metrics for One-path Model

Category	Precision	Recall	F-1 Score
Book	0.84	0.84	0.84
Home	0.95	0.96	0.96
Personal Care	0.83	0.83	0.83

Table 10. Performance Metrics for Two-path Model

Category	Precision	Recall	F-1 Score
Book	0.82	0.87	0.85
Home	0.95	0.96	0.96
Personal Care	0.86	0.80	0.83

As a result, the two-path model achieves slightly better scores than the one-path model. However, the two-path model is preferable due to its more stable loss values, as shown in Figure 11, indicating a lower likelihood of overfitting random patterns in the training dataset. The wide fluctuations in the one-path model's loss suggest it overfits some random patterns in the training batches, leading to incorrect predictions and higher loss for different batches. The significant gap in training losses indicates the discrepancies across batches.

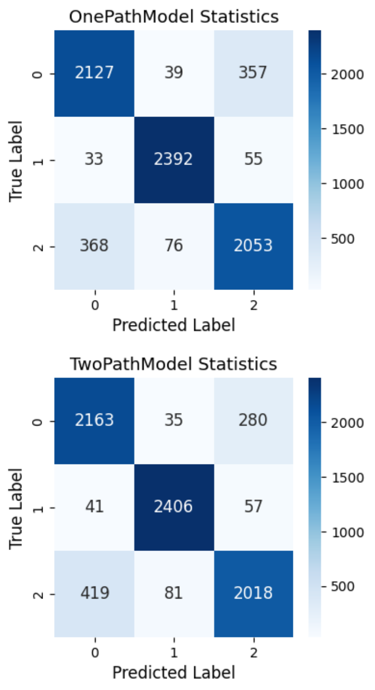

Fig. 12. Confusion Matrix for Neural Network

Additionally, the two-path model has 263,235 parameters, while the one-path model has 329,219 parameters, making the two-path model more computationally efficient in production. Due to the small dataset and the higher number of parameters, the one-path model can capture more patterns within the dataset but struggles to generalize them. In contrast, the two-path model, with fewer parameters, must find the best decision boundary that generalizes the training dataset by ignoring random patterns.

5 Conclusion

5.1 Performance Analysis

Precision, recall, and F-1 score metrics across models are similar using embedding vectorization and reduced dataset. On average, both neural network models are slightly better than the others (KNN and logistic models), especially for the hard to separate data in Book and Personal Care category. Neural network models also generalizes better to unseen data as the data size increases, where traditional machine learning models struggle. Conversely, traditional models like logistic regression and KNN are easier for stakeholders to interpret.

Using the full dataset of 3,000,000 data points, all the algorithms deployed in this study achieved very high scores above 95%. However, with the reduced dataset, the overall scores of these models dropped to about 85%. A smaller amount of data that's easier to predict results in lower statistical outcomes. Despite the lower scores, the models trained on the reduced dataset are more realistic and robust to unseen data.

The reason for this is that the density of common data points was reduced. This prevents the algorithms from favoring high-density data points that are easy to predict. By reducing the density of easily predicted data, the remaining dataset consists of data points that are more challenging for the models. This approach aims to balance the dataset, helping the models to generalize more effectively.

5.2 Training and Prediction Time

In production, we would need to retrain models on a periodic schedule. Fast training time is preferable because cross validation training can be time consuming to determine the best parameters for an algorithm.

Each model has its own strengths and weaknesses in terms of training and prediction time. KNN is the fastest during training, followed by logistic regression and neural networks. For predicting unseen data, the order is slightly reversed, with logistic regression being the fastest.

5.3 Stability and Robustness

Neural networks are generally more robust to outliers compared to KNN and logistic regression, primarily due to their ability to learn complex non-linear patterns and the availability of techniques to reduce the impact of outliers.

Logistic regression performs well when data is linearly separable, which is a significant disadvantage in this scenario. KNN can handle non-linear data but relies heavily on the dissimilarity function, which might not transform the data into a separable space. On the other hand, neural network models consist of multiple non-linear transformation layers to capture data variability.

5.4 Result Interpretation

All algorithms eventually reach a saturation point in accuracy because the inputs are compressed representation of the original data. Each word in title and feature is embedded using BERT; however, to produce a simple compact input, we average the words within the associated feature or title. Because a vector can be a linear combination of different bases (different set of spans), averaging all words in a feature text across different categories can lead to similar results, as shown by the intermixed yellow and purple points cluster in Figure 7.

Due to the limited and constrained amount of information by averaging data, the algorithm cannot fully utilize the semantic meaning for prediction. A solution is to use all the words in a title and associated features as input, taking the sequential processing advantage of LSTM and other advanced deep learning techniques to account for semantic word sequencing. However, deep neural network models are computationally expensive and may not be practical in production. Given that the main goal of this task is optimal classification without the need for interpretability, we recommend choosing a simple neural network model such as the two-path model for production.

6 References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 12 2013.

[2] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

[3] Vladimir Pestov. Is the k-nn classifier in high dimensions affected by the curse of dimensionality?, 2012.

[4] Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation - J DOC*, 60:503–520, 10 2004.

[5] Sara Sabour, Geoffrey E. Hinton, and Nicholas Frosst. Dynamic routing between capsules. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1:3859–3869, 10 2017.