Amazon ML Challenge

Product Browse Node Classification

- Aparna Sakshi
- Balaji Udayagiri
- Jalend Bantupalli

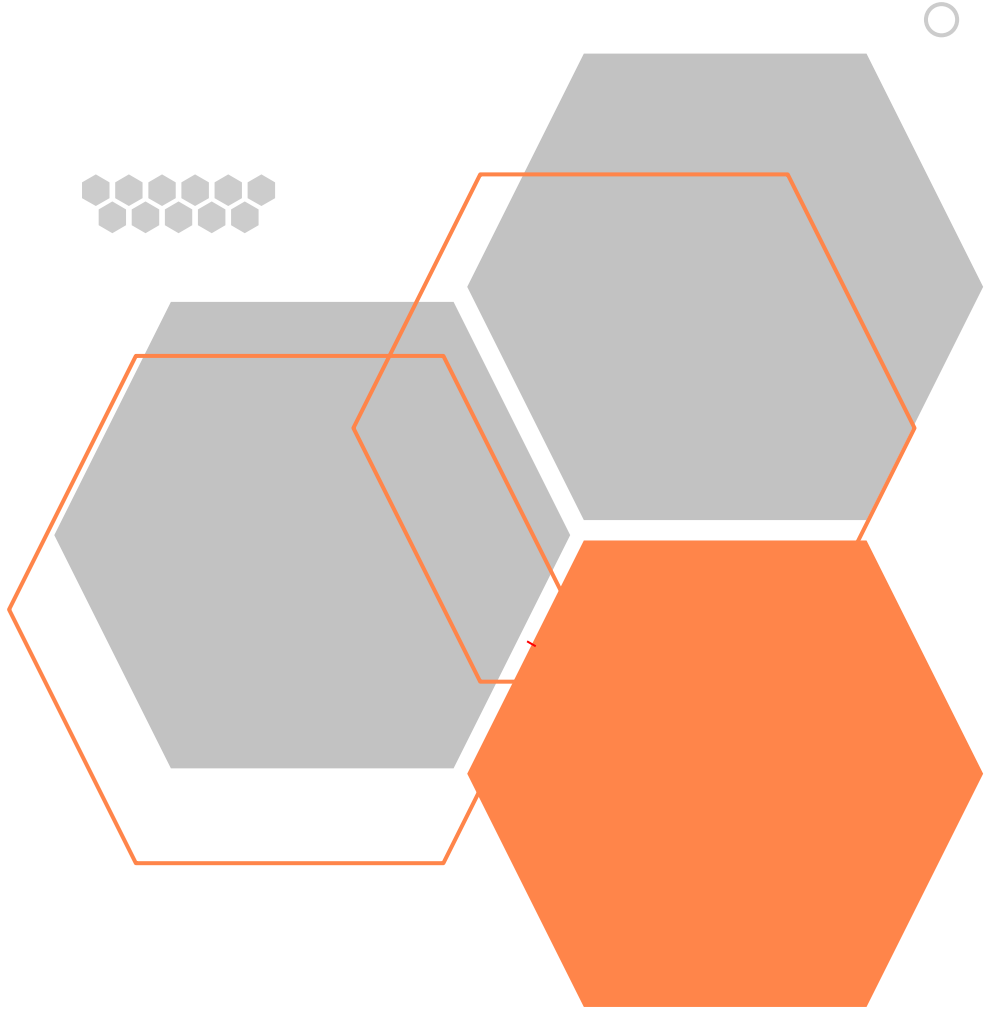
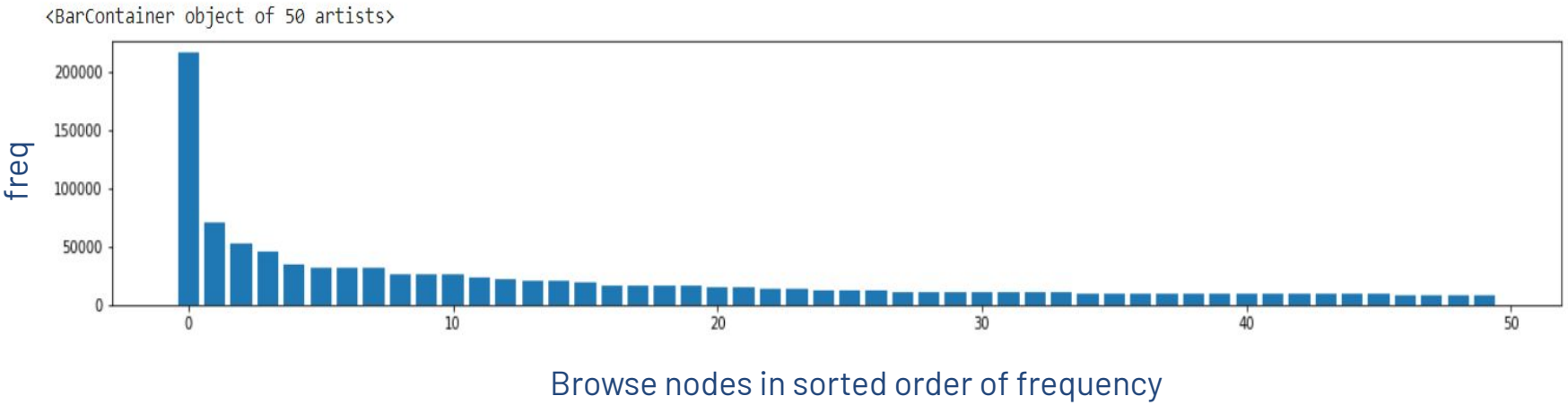


TABLE OF CONTENTS

DATASET DESCRIPTION

Training dataset size	Number of classes in Dataset	Overall Test Dataset Size	Input Features
2903024	9919	110775	Title,Description, Bullet_Points, Brand,



- This shows that the dataset is highly imbalanced.
- We observed that there are only 34 nodes with frequency greater than 10000.
- Browse Node with ID 1045 had the maximum frequency of 215698

PREPROCESSING

- Since it was not possible to load 2M rows at once, dataset was loaded and preprocessed in chunks, each chunk consisting of around 1 lakh rows.
- Text from TITLE, DESCRIPTION, BULLET_POINTS and BRAND was clubbed together and then preprocessed.
- Words in the test were tokenized, lemmatized, symbols were removed and the stop words were removed.
- Two files were created one for training consisting of 20 chunks and one for validation consisting of 10 chunks.

LEARNING APPROACH

- For this text classification task, hierarchical softmax was used.
- Hierarchical Softmax proves to be very efficient when there are a large number of categories and there is a class imbalance present in the data. Here, the classes are arranged in a tree distribution instead of a flat, list-like structure.

HIERARCHICAL SOFTMAX

- The construction of the hierarchical softmax layer is based on the Huffman coding tree, which uses shorter trees to represent more frequently occurring classes and longer trees for rarer, more infrequent classes.
- The probability that a given text belongs to a class is explored via a depth-first search along the nodes across the different branches. Therefore, branches (or equivalently, classes) with low probability can be discarded away.
- For data where there are a huge number of classes, this will result in a highly reduced order of complexity, thereby speeding up the classification process significantly compared to traditional models.

WORD N-GRAMS

- Using only a bag of words representation of the text leaves out crucial sequential information. Taking word order into account will end up being computationally expensive for large datasets.

- FastText incorporates a bag of n-grams representation along with word vectors to preserve some information about the surrounding words appearing near each word.

- This makes fastText an excellent tool for our problem statement

WHOA!

The training took around 12 min, and we able to achieve an accuracy of 74%