HGATT_LR

transforming review text classification with hypergraphs attention layer and logistic regression

Pradeepa, S.; Jomy, Elizabeth; Vimal, S.; Hassan, Md Mehedi; Dhiman, Gaurav; Karim, Asif; Kang, Dongwann

Published in:
Scientific Reports

DOI:
10.1038/s41598-024-70565-6

Published: 23/08/2024

Document Version
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Pradeepa, S., Jomy, E., Vimal, S., Hassan, M. M., Dhiman, G., Karim, A., & Kang, D. (2024). HGATT_LR: transforming review text classification with hypergraphs attention layer and logistic regression. *Scientific Reports*, *14*(1), 1-14. Article 19614. https://doi.org/10.1038/s41598-024-70565-6

scientific reports

OPEN

HGATT_LR: transforming review text classification with hypergraphs attention layer and logistic regression

S. Pradeepa[1], Elizabeth Jomy[2], S. Vimal[3], Md. Mehedi Hassan[4], Gaurav Dhiman[5,6,9], Asif Karim[7] & Dongwann Kang[8]✉

Text classification plays a major role in research such as sentiment analysis, opinion mining, and customer feedback analysis. Text classification using hypergraph algorithms is effective in capturing the intricate relationships between words and phrases in documents. The method entails text preprocessing, keyword extraction, feature selection, text classification, and performance metric evaluation. Here, we proposed a Hypergraph Attention Layer with Logistic Regression (HGATT_LR) for text classification in the Amazon review data set. The essential keywords are extracted by utilizing the Latent Dirichlet Allocation (LDA) technique. To build a hypergraph attention layer, feature selection based on node-level and edge-level attention is assessed. The resultant features are passed as an input of Logistic regression for text classification. Through a comparison analysis of different text classifiers on the Amazon data set, the performance metrics are assessed. Text classification using hypergraph Attention Network has been shown to achieve 88% accuracy which is better compared to other state-of-the-art algorithms. The proposed model is scalable and may be easily enhanced with more training data. The solution highlights the utility of hypergraph approaches for text classification as well as their applicability to real-world datasets with complicated interactions between text parts. This type of analysis will empower the business people will improve the quality of the product.

Keywords Amazon review dataset, Machine learning, Text classification, Hypergraph attention layer, HyperGAT, LDA, Logistic regression

Text classification, being the most basic task to be performed in natural language processing, has received ongoing research efforts due to its range of applications, including sentiment analysis[1], topic labelling[2], and disease detection[3]. Methods based on representation learning, such as CNN[4] and RNN[5], have been widely researched in recent years in response to the success of deep learning approaches.

GNNs are a form of neural network designed to function on graph-structured data, which uses nodes and edges to represent objects and their relationships. They have gained popularity due to their capacity to handle complicated object interactions, making them appropriate for a variety of applications. GNNs are classified into numerous types based on their architecture, including spectral-based and spatial-based models. Graph neural networks have proven to be excellent at tasks with complex relationship structures and can preserve global structure information in graph embedding[6].

The use of graph neural networks (GNNs) as a powerful text categorization tool has attracted a lot of interest in the field of natural language processing (NLP)[7]. However conventional GNN algorithms struggle to capture complex word interactions, which is essential for effectively capturing context in text documents. A novel strategy

[1]Department of Information Technology, School of Computing, SASTRA Deemed University, Thanjavur, Tamilnadu 613401, India. [2]Department of Computer Science and Engineering, School of Computing, SASTRA Deemed University, Thanjavur, Tamilnadu 613401, India. [3]Department of Artificial Intelligence and Data Science, Sri Eshwar College of Engineering, Coimbatore 641202, India. [4]Computer Science and Engineering Discipline, Khulna University, Khulna 9208, Bangladesh. [5]Centre of Research Impact and Outcome, Chitkara University, Rajpura, Punjab 140417, India. [6]MEU Research Unit, Middle East University, Amman, Jordan. [7]Faculty of Science and Technology, Charles Darwin University, Darwin, NT 0909, Australia. [8]Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea. [9]Division of Research and Development, Lovely Professional University, Phagwara, India. ✉email: dongwann@seoultech.ac.kr

that uses document-level hypergraphs for text document modelling is presented in order to overcome this difficulty. By integrating hypergraph structures, the suggested model, HyperGraph Attention Layer with Logistic Regression (HGATT_LR), improves upon the capabilities of traditional Neural Network. With the help of this creative framework, heterogeneous high-order contextual information about each word may be captured, improving the expressiveness of the model while preserving computing efficiency.

HGATT_LR offers a distinctive and efficient approach to text representation learning by integrating Logistic Regression (LR) to further improve the classification process, in contrast to other algorithms like HyperGAT. Enhanced interpretability and fine-grained analysis of text classification results are made possible by the incorporation of LR, which sets HGATT_LR apart as a flexible and potent tool for a range of NLP applications, including sentiment analysis, opinion mining, and customer feedback analysis.

The rest of the paper is formatted as follows. In "Related works", a thorough analysis of relevant literature, covering current methods in graph neural networks, hypergraph-based models, and text categorization is presented. The data collecting procedure, feature extraction methods and description of the HGATT_LR model are all covered in length in "Methodology and description", along with our approach. The Amazon Review dataset, model testing and training, as well as the classification metrics and confusion matrix for HGATT_LR, are all covered in "Experimental results" of the experimental results.

Related works

Several approaches to sentiment analysis and text data classification have been investigated in the field of text classification, particularly in the context of Amazon reviews. To extract sentiment polarity from textual data, several research has used machine learning methods such as Naive Bayes, SVM, and RNN. Pre-trained language models have drawn a lot of interest recently because of their capacity to capture intricate semantic links and contextual information.

Furthermore, by adding reward signals during training, reward Learning techniques have been employed to improve classification accuracy. Thus, there is a need to fill this research by developing a one-of-a-kind solution that leverages the strength of these approaches to provide accurate sentiment polarity analysis in real time for user-inputted text evaluations. A large-scale Amazon review database is used to polarize it and attain appropriate precision.

The study Reinforcement Learning Approach seeks to classify Amazon reviews into binary and maybe multiclass categories using Reinforcement Learning and a pre-trained BERT model. The research will result in the creation of a web application that will use the proposed model to deliver real-time sentiment polarity to a user-inputted text review[8]. The embedding layer derives the word embedding/dense representation for the 2500 words by training the model. The model Sentiment Analysis through LSTMs is then enhanced with LSTM and Dense layers. The LSTM cell is in charge of making contextual inferences and determining whether a statement is favourable or negative. The probabilities for each class are output by the Dense layer[9].

A short text classification approach based on a hypergraph convolution network can capture the complicated high-order associations of words in a short text in a flexible manner. The approach represents a single text as a network of hyperedges, with all words connected. The feature representation of words and brief text is then learned using a hypergraph convolution network. Finally, the classification model is utilized to classify short text[10]. Graph neural networks (GNNs) have recently attracted more interest in the research community because of their promising outcomes on this standard challenge. A principled model—hypergraph attention networks (HyperGAT)—for text representation learning was proposed that can achieve more expressive power with less CPU consumption. Extensive testing on several benchmark datasets shows that the suggested approach is effective at text classification[7].

An essential area of study is the fusion of LSTM and other architectures. The textual entailment problem[11] was solved using the Child-Sum Tree-LSTM, which is simple and generalizable. The documentation showed[12] that LSTM outperforms current state-of-the-art approaches in predicting protein subcellular localization given only the protein sequence. Convolutional filters and an attention mechanism were used to concentrate on specific areas of the protein, which the LSTM enhances. In the study Dimensional sentiment analysis using a regional CNN-LSTM[13] suggested a regional CNN-LSTM model predict text valence-arousal (VA) ratings that consist of two parts: regional CNN and LSTM[13]. Table 1 depicts the overview of the datasets and the accuracy of the text classification in the review dataset.

The regional CNN divides a text input into portions to collect and weigh important emotional data according to each region's contribution[14]. Sentimental analysis is a rapidly expanding study field in which text mining plays an important part in assessing customer feedback. Sentimental analysis of Amazon customer's review comments[15] suggests a hybrid approach that combines technologies like DL, ML, and NLP to analyze customer evaluations and ratings. Using ML and natural language processing, sentiment analysis is performed on the cleaned dataset. The most effective classifier algorithms, including SVM and NB with NLP, are used in that presentsto carry out sentiment analysis. Xiaoming Shi and Ran Lu[16] present an attention-based bidirectional hierarchical LSTM network model. The suggested method outperforms existing machine learning methods and semantic categorization methods, according to experiment findings on Yelp, IMDB, Yahoo Answer, and Amazon review datasets. The categorization and stability accuracy rates outperform competitors and attain state-of-the-art outcomes.

Methodology and description

The process for creating the HyperGraph Attention Layer with Logistic Regression (HGATT_LR) model is described in this section. The methodology begins with data collecting and uses techniques such as TF–IDF and LDA for feature extraction. The steps taken to enhance text classification performance and capture high-order

Title of paper	Dataset used in the paper	Model used	Results accomplished
Amazon Reviews Sentiment Analysis: A Reinforcement Learning Approach (2020)	Amazon product data is a subset of 142.8 million Amazon analysis datasets that Stanford professor, Julian McAuley, has made available	LSTM as Policy Network and BERT as Classification Network	LSTM—95% BERT—96.3% LSTM with PPO—98%
Sentiment Analysis through LSTMs (2018)	Amazon products reviews	LSTMs	LSTM—82% LSTM with Word2vec—94.4% LSTM—98%
Short Text Classification via Hypergraph Convolution Network (2021)	Movie reviews with only one sentence per review	STCH (short text classification hypergraph	STCH—60.573 (has highest when the same dataset is compared with SVM, CNN & LSTM)
Be More with Less: Hypergraph Attention Networks for Inductive Text Classification (2020)	20NG R8 R52 Ohsumed MR	HyperGAT	20NG—86.62% R8—97.97% R52—94.98% Ohsumed—69.90% MR—78.32%

Table 1. An overview of the datasets and accuracy.

contextual information from Hypergraph attention layer and then classified using Logistic regression. The main contribution of the proposed model is.

- Collect the Amazon review data from the Kaggle and preprocess using Term Frequency and Inverse Document Frequency. The features are extracted using Latent Dirichlet Allocation.
- The essential keywords are identified using Hypergraph Neural Network (HGNN). Finally, the output is then used in any classifier and in this case, a logistic regression classifier is considered.

This study aimed to anticipate consumer reviews for a specific product using machine learning techniques. The Amazon dataset is taken from Kaggle and consisted of product reviews submitted by consumers. Figure 1 shows the proposed architecture for text classification in the review data set.

Data collection

The Amazon review dataset is a previously retrieved dataset from Kaggle[17] that includes textual consumer reviews and labels sorted based on their sentiment. The dataset includes textual consumer reviews on any Amazon product, as well as labels categorized by sentiment, such as pos (positive) and neg (negative) based on customer's comments. Table 2 depicts the sample Amazon review dataset in excel format.

Data preprocessing

The next stage in any deep learning system is pre-processing, which involves breaking down text data into smaller tokens such as words or concepts. Thus, using modules from the natural language toolkit (nltk), the text is turned into a clean and consistent text collection[18]. To do this, the nltk tokenize module is used, and the stem module strips prefixes and suffixes to reduce them to stem forms. The retrieved data contains a huge number of frequently used irrelevant English terms that must be deleted to focus on the keywords. Numpy, NLTK, re, string and spacy libraries include built-in functions and tools to aid with data preprocessing operations. Any URLs or HTML tags that are extraneous to the research are deleted from the text data. Characters that are not required, including non-alphanumeric characters and extra spaces, are eliminated from the text data.

All formatting, including single and double quotes and backslashes, is removed from the text data. It receives a list of documents, each consisting of a list of sentences, as input. Low-frequency terms, stopwords, and lemmatized words are eliminated. A list of clean documents with clean sentences in each is the end output.

Feature extraction

Feature extraction in textual data entails transforming raw text into numerical representations appropriate for machine learning techniques. Tokenization is often used to break down text into words or tokens, stopword removal, and normalization using techniques such as lemmatization or stemming, followed by vectorization using Bag-of-Words (BoW), *TF–IDF*, or word embeddings.

Term Frequency–Inverse Document Frequency (*TF–IDF*)[19] is a numerical statistic used in information retrieval and text mining to determine the importance of a term within a document or a corpus of documents. It takes into account both the frequency of a term in a document (Term Frequency) and its rarity across all documents (Inverse Document Frequency). The final *TF–IDF* score for a term in a document combines the *TF* and *IDF* values, assigning higher weights to terms that appear frequently in a specific document (*TF*) but are rare in the overall corpus (*IDF*).

In the *TF–IDF* approach, each document's feature set is initially built by gathering all of its individual words. Furthermore, the *TF–IDF* score for each word in each document is determined. Additionally, the *TF–IDF* scores of each word in a document are used to order them[19]. The created feature set is then used to represent each document in the corpus. The TfidfVectorizer class from the scikit-learn library is used for feature extraction to turn text into numerical feature vectors. The tfidf Vectorizer method computes *TF–IDF* of each term in the corpus.

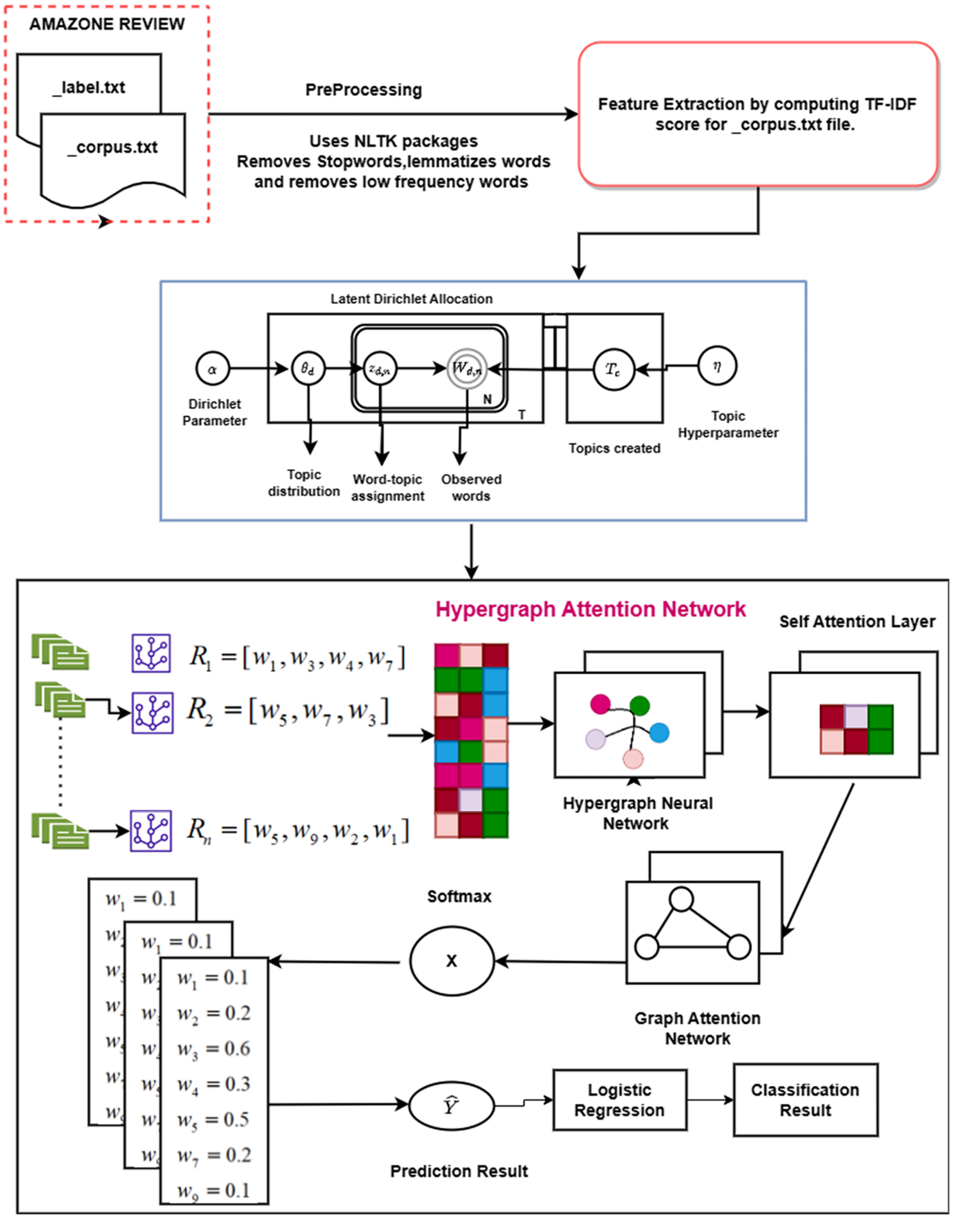

Fig. 1. The architectural overview of proposed method (HGATT_LR).

S. no.	Label	Final_text
0	pos	Amazing even non-gamers soundtrack wonderful paints scenery mind well would recommend even people who dislike video game music the worst keyboarding ever takes fresher step grates guitars soulful orchestras will impress anyone bothers to listen

Table 2. Sample Amazon review dataset in Excel format.

In IDF, t is the term (word) to be measured and $f_{(td)}$ is number of times the t occurred in the document d. $\frac{f_{(td)}}{\sum_{t\prime \in d} f_{(t\prime d)}}$ represents the number of terms in the document. Total number of documents (N) and $df_{(t)}$ shows the number of document (d) holds the term (t) in the corpus[20].

$$IDF(t) = log \frac{N}{df_{(t)}}$$

$$tf(t,d) = \frac{f_{(td)}}{\sum_{t\prime\in d} f_{(t\prime d))}}$$

$$tfidf(t,d,D) = tf(t,d) * IDF(t)$$

The more relevant or important a term is, the higher its TF–IDF score is; as a term becomes less relevant, its TF–IDF score will decrease until it is zero.

Each article in the corpus is converted into an integer vector by the TfidfVectorizer algorithm by expressing every word as a vector element, with the value of the element denoting the document's TF–IDF score. These feature vectors are then used to train machine-learning models for text classification.

Topic modeling with LDA

LDA is a component of the probabilistic topic model, when compared to other models, LDA has stronger descriptive strength and better semantic criteria. Preprocessing the text input, making a dictionary and bag-of-words representation of the data, and creating an LDA file are all steps in the process. Training the LDA model on the bag-of-words data is the subsequent process that is carried out, and then saving the model to a file as a pickle file. The resulting LDA file can then be used to perform topic modelling on new data. We convert the tokenized documents into gensim's bag-of-words format, after having a dictionary, which depicts each article as a sparse vector of word counts. Then, create an LDA Model object and pass in the bag-of-words data, dictionary, and number of topics to generate. This is an iterative technique that increases the topic quality with each pass. This generates an LDA file, which we can subsequently import and use to do topic modelling on new data.

The k of the Dirichlet distribution is deemed acknowledged and set. The parameterization of word probabilities of a $k \times V$ matrix β where, $\beta_{i,j} = P(w_j = 1|z^i = 1)$ which we treat as a fixed amount that needs to be predicted for the time being[21]. The remainder of this section does not depend on the Poisson assumption, and more accurate document length distributions may be employed as necessary. In addition, N is independent of every other variable that produces data (θ and z). It is therefore an ancillary variable, and we will typically ignore its randomness in the subsequent development.

$$P(\theta|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{k}\alpha_i\right)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\theta_1^{\alpha_i-1}\cdots\theta_1^{\alpha_k-1}$$

The variables α and β, the combined allocation of a subject mixture θ, a set of N subjects z, and a set of N terms w.

$$P(\theta,z,w|\alpha,\beta) = p(\theta|\alpha)\prod_{n=1}^{N}p(z_n|\theta)p(w_n|z_n,\beta)$$

{'wa': [0, 1], 'great': [0, 1], 'good': [0, 1], 'like': [0, 1], 'buy': [0], 'time': [0, 1], 'product': [0], 'work': [0], 'dont': [0], 'cd': [0], 'book': [1], 'movie': [1], 'read': [1], 'story': [1], 'ha': [1]} Words stored in the _LDA.p (Pickel file) is shown in Fig. 2. These words are common concepts identified in the dataset and are classified according to their context. The digits 0 and 1 show which topic each word belongs to .LDA is used to identify and categorize topics in the dataset, and the number of topics is set to two to reflect general product reviews and media content reviews. For example, Topic 0 contains words like "buy," "product," "work," and "cd," whereas Topic 1 contains words like "book," "movie," "read," and "story." This allows to descriptionof each document as a distribution over these themes, lowering data dimensionality and highlighting the most important features for classification, hence improving overall classification performance.

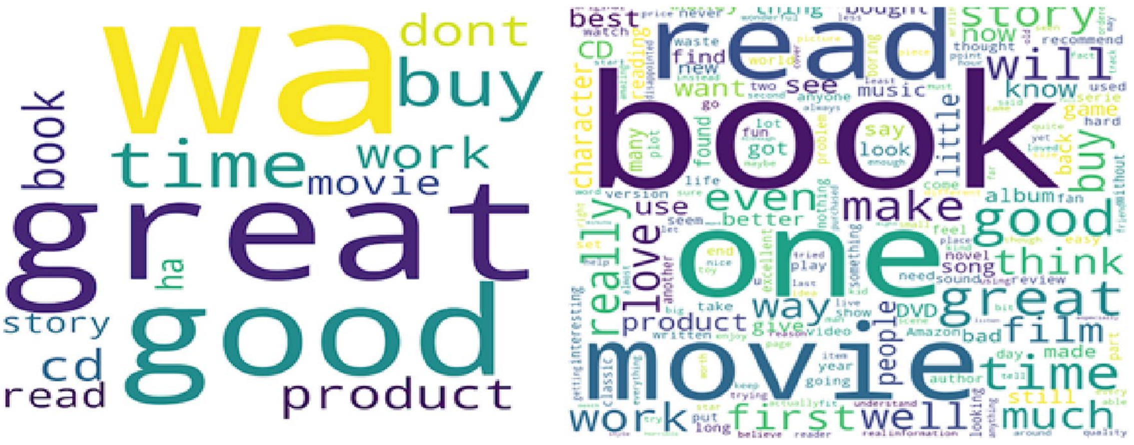

Fig. 2. Wordcloud of dataset and content in _LDA.p (pickel file), right to left.

Implementation of hypergraph neural network

The purpose of hypergraph attention is to learn a dynamic incidence matrix, and thus a dynamic transition matrix, which can disclose the inherent relationship between vertices more clearly[22].The attention module gives a probabilistic model that assigns non-binary real values to characterize the degree of connectivity between vertex and hyperedge[23]. The fundamental advantage of hypergraph attention layers is that they may record higher-order interactions between data points, which is especially beneficial in activities with complicated and interdependent data point relationships. Figure 3 shows the comparison of a Hypergraph Neural Network (HGNN) and Dropout HGNN.The number of input and output features relates to the size of the neural network model's input and output layers. These parameters determine the input and output data dimensions that the model can handle as well as the total neurons in every layer. The dropout rate is a regularization strategy used to prevent model overfitting[24]. During training, it randomly removes a certain number of neurons, which helps to lessen the model's dependency on certain features and enhances its generalization capacity.

A hypergraph attention layer is utilized in this because of its ability to simulate complex interactions between diverse elements in a dataset. The concept of attention mechanisms is used in hypergraph attention layers to learn the relevance of nodes and hyperedges in a hypergraph. Attention mechanisms add weights to each node and hyperedge in the hypergraph, indicating their importance in the data representation. Weights are learned during training based on the relevance of each node and hyperedge to the task at hand. The dropout rate is a regularization approach that removes a particular number of neurons at random during training to prevent model overfitting[25].

Attention mechanism for document classification

An attention mechanism is employed to give a different focus to the information outputted from the hidden layers[26]. The basic idea behind Hypergraph Neural Network (HGNN) with attention mechanism is to describe the document using a hierarchical graph structure, with each level of hierarchy represented as a separate graph. Each graph's nodes represent the components at that level, while the edges reflect the relationships between them. The overall HGNN with attention mechanism architecture for document categorization incorporates numerous layers of graph convolutional networks and attention mechanisms, each capturing a distinct level of hierarchy in the content.

Hypergraph neural network with attention mechanism (HGATT)

Hypergraph Neural Network (HGNN) uses the attention mechanism to determine the relevance of each node in the graph representation[22]. Attention mechanisms are used in natural language processing tasks to learn the relative relevance of distinct parts of the input sequence. This allows the model to focus on the most significant parts of the document. The graph neural network concept generalizes the convolution operation to graph data. To use the attention mechanism, a weight vector representing the relevance of each node in the graph is learned. This weight vector is computed based on the features of the node and its neighbors, and is updated iteratively based on the attention scores computed for the neighboring nodes. This allows the model to capture contextual information of the node in the graph.

The HGNN with Attention model is implemented by the code which defines its component parts. A linear layer for prediction, an embedding layer, layer normalization layers, and layers for attention-based graph neural networks (GNNs) with hypergraph attention are among these components. The GloVe word embedding model, which has been trained beforehand, is also utilized to get word vectors from a given vocabulary dictionary and produce a matrix of word vectors that match the dictionary terms. The first row is all zeros, which is used to

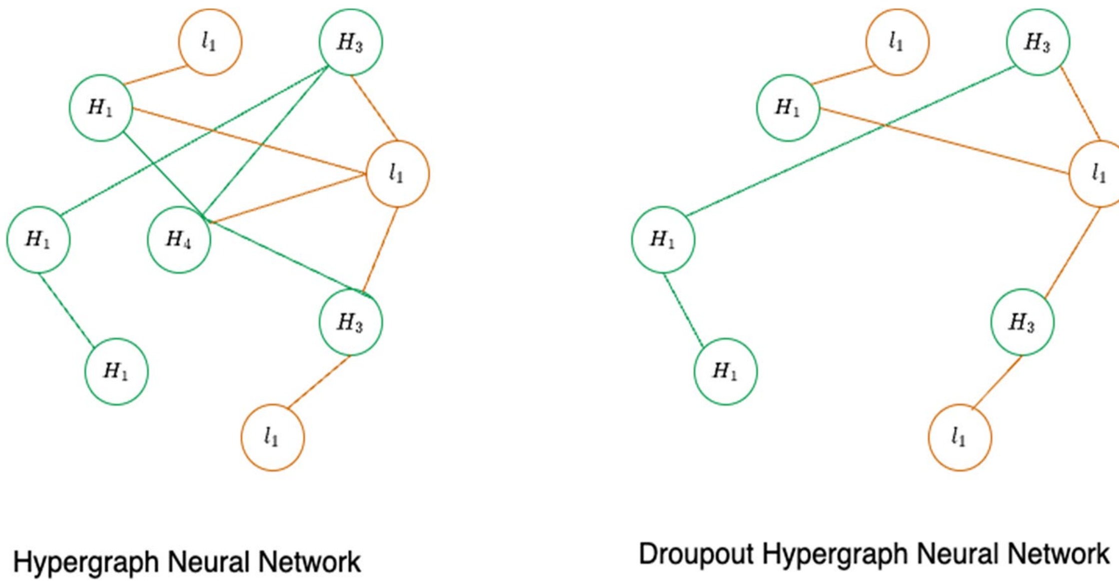

Fig. 3. Comparison of a hypergraph neural network and dropout HGNN.

represent unknown words. Algorithm 1 shows the Hypergraph Attention layer construction. Figure 4 shows the Hypergraph Attention Mechanism.

```
Inputs:
        Hypergraph Incidence Matix ($\boldsymbol{H} \in \boldsymbol{R}^{N\times M}$) : N nodes, M hyperedges
        Weight Matrix ($W \in R^{F\times F'}$)
        Attention vector : $a \in R^{2F'}$
        Non Linear Activation function ( ReLU)$\sigma$
        Input node features ( F features per node)$X \in \boldsymbol{R}^{N\times F}$ : F feature per node
Output:
        $H^{(out)} \in \boldsymbol{R}^{N\times F'}$
Step:
    1. Compute linear transformation of the input node features: $X' = XW$
    2. Compute attention coefficients for each node-hyperedge. Here, $i$ represents node and $j$
       represents hyperedge.
           $e_{ij} = LeakyReLU(a^T.[X_i'||X_j')$
       Here, || denotes the Concatenation, and $X_i'$ shows the feature vectors for node $i$ and $X_j'$
       represents the feature vectors for hyperedges $j$.
    3. Normalize the attention coefficient using the softmax function
           $\gamma_{ij} = \frac{\exp(e_{ij})}{\sum_{k\in N(j)} \exp(e_{ik})}$
       $N(j)$ represents the number of nodes connected to hyperedge.
    4. Aggregate the information from neighboring nodes and hyperedges, weighted by the
       attention coefficients:
           $H_i^{(out)} = \sigma(\sum_{k\in N(j)} \alpha_{ij}X_j')$
```

Algorithm 1: Hypergraph_Attention_Layer (HGNN_ATT).

Iteratively training a word embedding model on batches of data, evaluating its performance on a validation set, and assessing its capacity to generalize to new data on a test set comprise the training and testing technique. It reads the Amazon corpus and its labels from files using two arguments: dataset and LDA. It then constructs a vocabulary dictionary and a label dictionary in order to map words and labels to integer indices. Finally, it computes class weights for the training set and returns the preprocessed data as well as any additional information.

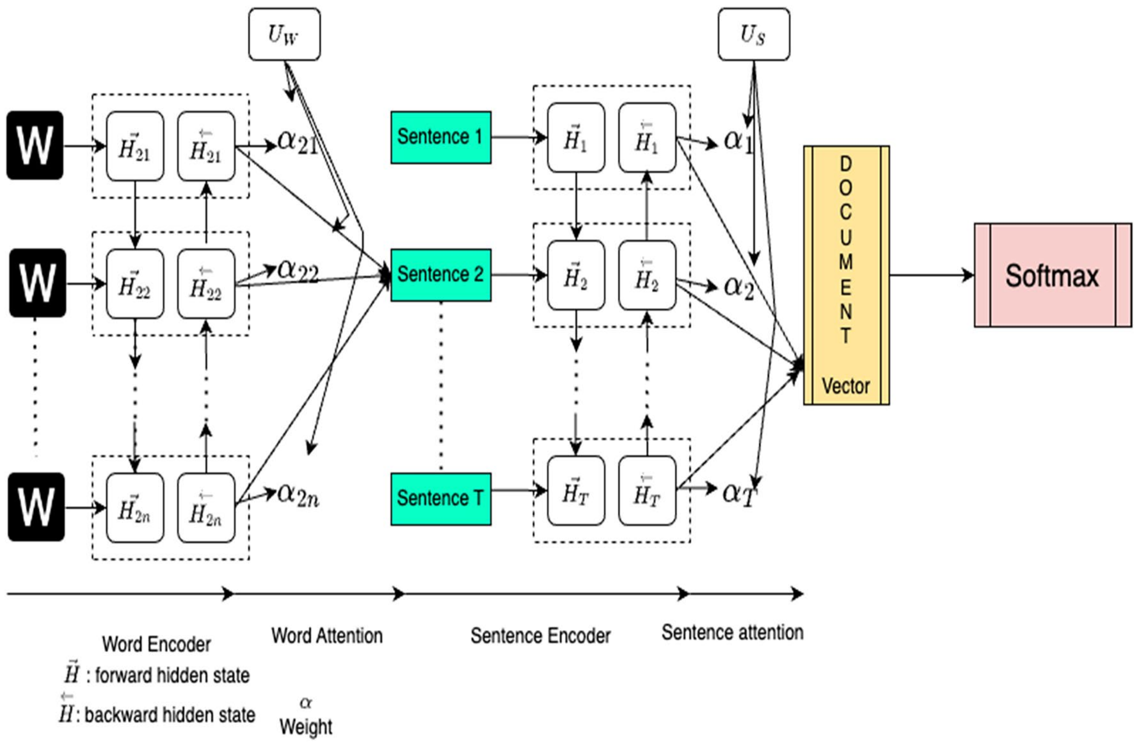

Fig. 4. Hypergraph attention mechanism.

The train_model function, which performs forward and backward propagation on each batch of training data, is used to train the model for a defined number of epochs (provided by theepoch parameter). The model is evaluated on a validation set throughout each epoch using the test_model function with the validate argument set to True. The validation set is a subset of the training data used to test the model's performance on unknown data and determine if it is overfitting. Following training, the model's final performance is evaluated on a test set with the validate parameter set to False.

The train_model and test_model functions load training and testing data in batches using PyTorch's Data-Loader class. The train_model function computes the loss for each batch and uses backpropagation to adjust the model's weights. This framework is applied to the test set via the test_model function, which then determines how accurate these predictions are.

HGATT with logistic regression model (HGATT_LR model)

Text classification is the process of classifying a document. Logistic Regression (LR) is used for text classification, as it can provide probabilities and extend to multi-class classification problems. It also follows the same principles as linear regression. Text classification is divided into two categories: topic-based text classification and text genre-based text classification. Topic-based text categorization classifies documents according to their topics[27]. Intuitively, the task of categorizing a material under a predefined category is known as text classification[28].

Logistic Regression is utilized for text classification in this case. The main advantages of LR are that it can naturally provide probabilities and extend to multi-class classification problems[29,30].Another advantage is that most of the methods used in LR model analysis follow the same principles used in linear regression[31].

In the HGATT_LR model, the TfidfVectorizer class converts text data to numerical data using the vectorize and split methods. The data is preprocessed using the prepare_data_for_train function, which lowercases the text, eliminates URLs, HTML tags, punctuation, digits, and stopwords, and tokenizes it with Spacy. It accepts input data and returns preprocessed data. Algorithm 2 demonstrates the Logistic Regression method used in HGATT_LR architecture.The base method is used to train the HGATT_LR model and evaluate its performance. It uses the prepare_data_for_train, split, and vectorize methods to preprocess the data, split it into train and test sets, and transform it. The performance method is used to display the performance of the model. The test_sample method is used to predict the sentiment of a single review. It takes in the review text, the TfidfVectorizer object, and the trained model and preprocesses it using the clean_df method.

```
Input:
    $\boldsymbol{X} \in \boldsymbol{R}^{m\times n}$ Input feature matrix ( m samples, n features)
    Target Vector $\boldsymbol{y} \in \{\boldsymbol{0},\boldsymbol{1}\}^{\boldsymbol{m}}$
    Learning rate $\emptyset$

Output:
    Weight Vector :$\boldsymbol{w} \in \boldsymbol{R}^{\boldsymbol{N}}$
    Bias Term : b$\in \boldsymbol{R}$

  1. Intialize the parameters $\boldsymbol{w}$ =0 and $\boldsymbol{b}$=0.
  2. Define sigmoid function $\boldsymbol{\sigma}(\boldsymbol{z}) = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{exp}(-\boldsymbol{z})}$
  3. Define the hypothesis function $\boldsymbol{h}(\boldsymbol{x}_i) = \boldsymbol{\sigma}(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{x}_i + \boldsymbol{b})$
  4. Define the cost function $\boldsymbol{J}(\boldsymbol{w},\boldsymbol{b}) = \frac{\mathbf{1}}{\boldsymbol{m}}\sum_{i=1}^{m}[\boldsymbol{y}_i\boldsymbol{log}(\boldsymbol{h}(\boldsymbol{x}_i)) + (\mathbf{1}-\boldsymbol{y}_i)\boldsymbol{log}(\mathbf{1}-(\boldsymbol{h}(\boldsymbol{x}_i)))]$
  5. Compute the gradients
         $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{w}} = \frac{\mathbf{1}}{\boldsymbol{m}}\sum_{i=1}^{m}(\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{y}_i)\boldsymbol{x}_i$
         $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{b}} = \frac{\mathbf{1}}{\boldsymbol{m}}\sum_{i=1}^{m}(\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{y}_i)$
  6. Update the parameters w and b
         $\boldsymbol{w} = \boldsymbol{w} - \emptyset\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{w}}$
         $\boldsymbol{b} = \boldsymbol{b} - \emptyset\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{b}}$
```

Algorithm 2. Logistic regression.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Experimental results

The findings provide useful insights and outcomes for using the proposed strategy for text categorization on the Amazon dataset. The model's performance was evaluated using rigorous experimentation and evaluation in terms of accuracy, precision, recall, and F1-score. These measurements are critical in determining the success and dependability of the suggested solution. The results analysis and interpretation have important implications for future research and practical applications in natural language processing and text classification. In this section, some key results are presented that show the effectiveness of HyperGAT.

Amazon review dataset

The Amazon Review dataset consists of 4903 of positive reviews and 5097 of negative reviews. The aim is to classify this dataset using hypergraph classification. Figure 5 gives the overview of the total number of reviews classified as a positive and negative label.

According to the dataset's statistics on sentence length, the shortest sentence length is 0 characters and the longest is 127 characters. Each entry contains an average of 8.51 characters, with a minimum of one sentence and a maximum of twenty sentences. The collection contains a total of 7629 words. The dataset is separated into two categories: negative and positive, with 5097 negative sentences and 4903 positive words. Academics and professionals can better comprehend the distribution and properties of text data thanks to these statistics, which offer a broad overview of the dataset's qualities. Table 3 shows the dataset statistics.

Testing and training using hypergraph neural networking

Iteratively training a word embedding model on batches of data, evaluating its performance on a validation set, and assessing its ability to generalize to new data on a test set comprise the training and testing technique. Intel Core i5-1035G1 (16 GB RAM), Python 3.8, PyTorch 1.8.0, CUDA 11.0, cuDNN 8.0.5, and relevant libraries such as scikit-learn, NumPy, NumPy, and Pandas are utilized for the development of proposed model. The training is conducted with batch size 32. While the test_model function applies our framework to the test-set and evaluates the precision of these predictions, the train_model function computes the loss for each batch and back propagates the model's weights. Acquired a validation accuracy of 83.87 and test accuracy of 84.38. The validation accuracy is calculated during the training phase using a separate validation dataset and serves as a measure of how well the model learns from the training data without overfitting. Similarly, test accuracy is computed using a completely independent test dataset and represents the model's ability to generalize to new, previously unseen data.

Table 4 shows that the model performed well by correctly predicting the classes of the validation examples. On the test dataset, the model attained an accuracy of 84.38%, indicating that it generalized well from the training data to unseen data. The model's total accuracy, which combines validation and test results, suggests that the model performed consistently well in categorizing cases across diverse datasets. The model's accuracy was

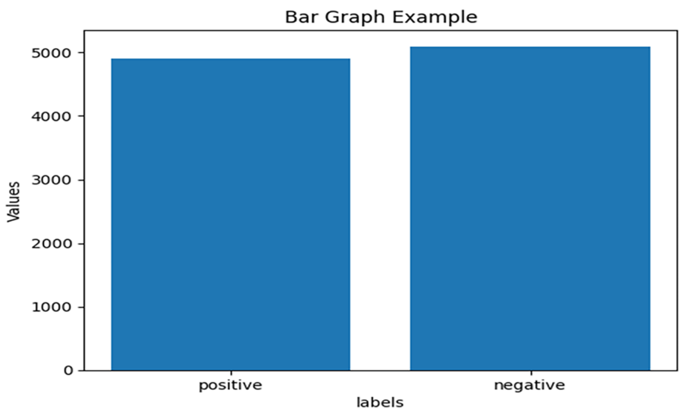

Fig. 5. Bar graph to compare the number of reviews as positive and negative.

Minimum length of sentence	**0**
Maximum length of sentence	**127**
Minimum number of sentence	**1**
Maximum number of sentence	**20**
The average length of sentences	8.510052500915714
Average number of sentences	4.9142
Counter	
1. Negative (neg)	1. 5097
2. Positive (pos)	2. 4903
Total number of words	7629
Total number of categories	2

Table 3. Dataset statistics. Significant values are in bold.

Parameters	Precision	Recall	f1-Score
neg	84.24	83.40	83.82
pos	83.56	84.40	83.98
Accuracy	84.38		
Validation accuracy	83.87		
Test accuracy	84.38		

Table 4. Hypergraph text classification model; validation and test datasets.

83.42%, indicating that it had a low rate of false positives. The model has an 83.40% recall, meaning that it correctly detected a large percentage of positive events. The model's F1-score was 83.40%, showing a solid mix of precision and recall. Figure 6 shows the accuracy for the proposed model.

The findings show that the text categorization model performed well in general, with good accuracy, precision, recall, and F1-score values. These metrics reflect the model's ability to reliably identify cases as well as capture the underlying patterns and information in the dataset.

Hypergraph with logistic regression classification (HGATT_LR model)

The classification model achieved high precision, recall, and F1-score values for both categories neg and pos, with consistent performance across the dataset, as shown in Table 5. The model's accuracy of 88% suggests that it accurately classified a vast number of the occurrences. Higher precision numbers imply more accurate forecasts, with 88% of "neg" predictions being negative and 85% of "pos" predictions being positive.

According to the recall values for the "neg" and "pos" categories, the model accurately detected 86% and 88% of the positive instances, respectively. The F1-score is 0.88, indicating that precision and recall are well balanced. Support gives insight into the distribution of instances across categories, with 516 instances for the "neg" category and 484 instances for the "pos" category. Acquired an accuracy of 88% on the Amazon with Logistic regression classification.

Confusion matrix

A confusion matrix can be used to assess the performance of a classification model. It displays the model's true positive, true negative, false positive, and false negative predictions. The confusion matrix compares the actual target values with those predicted by the machine learning model[32]. It is used to assess the performance of

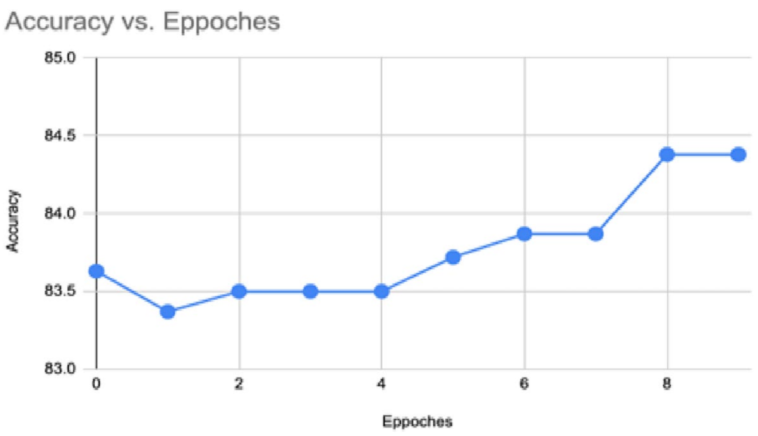

Fig. 6. Accuracy distribution during each epochs.

Parameters	Precision	Recall	f1-Score	support
neg	88	86	88	516
pos	85	88	87	484
Accuracy			88	1000
Macro avg	87	87	87	1000
Weighted avg	87	87	88	1000

Table 5. Hypergraph with logistic regression model (HGATT_LR).

classification models, which predict a category label for each occurrence of an input. The matrix for binary classification will be a 2 × 2 table, whereas the matrix shape for multi-class classification will be equal to the number of classes, i.e. nXn for n classes. Two models' evaluation metrics can be compared to decide which is preferable. If model A has a higher accuracy than model B, then model A is superior. Similarly, if model A has a higher precision, recall, and F1 score than model B, we can say that model A is superior in that metric. In this, the real targets are contrasted with those that the ML model anticipated.[32].

1. True Negative (TP): N negative class data points were appropriately categorized by the framework[32].
2. False Positive (TN): The framework misclassified M data points from the negative class to belong in the positive class[32].
3. True Positive (FP): N positive class data points were appropriately categorized by the framework[32].
4. False Negative (FN): The framework misclassified M data points from the positive class to belong in the negative class[32]. Here Fig. 7 shows the confusion matrix for Hypergraph and HGATT_LR Model

$$precision(P) = \frac{TP}{TP + FP}$$

$$Recall(R) = \frac{TP}{TP + FN}$$

$$F1 - Score(F1) = 2\frac{PR}{P + R}$$

$$Accuracy(A) = \frac{TP + TN}{TP + TN + FP + FN}$$

ROC curve and precision–recall curve

When dealing with imbalanced datasets or when the emphasis is on the positive class, the PR curve comes in handy. It provides a more thorough assessment of our framework's performance than accuracy alone, particularly when the distribution of good and negative examples is uneven. Our framework's performance can be analyzed using precision and recall by analyzing the PR curve and computing the AP score, and then make educated decisions regarding the classification threshold depending on the desired balance between the two.

The area under the ROC curve (AUC-ROC) is a popular statistic for assessing a binary classification model's performance. It quantifies the model's ability to differentiate between positive and negative events across all threshold values. AUC-ROC of 1 represents a flawless classifier, whereas 0.5 implies a random classifier. The overall effectiveness of the classification model and select an ideal threshold by analyzing the ROC curve and computing the AUC-ROC. The model acquired AUC-ROC of 0.95 thus suggesting it being closed to flawless classifier. The higher the AUC-ROC score and the closer the ROC curve is to the top-left corner, the better the model's ability in differentiating between positive and negative cases. Figures 8 and 9 shows the Precision -Recall curve and ROC curve for the HGATT_LR model. Table 6 shows the record of parameters for Hypergraph model.

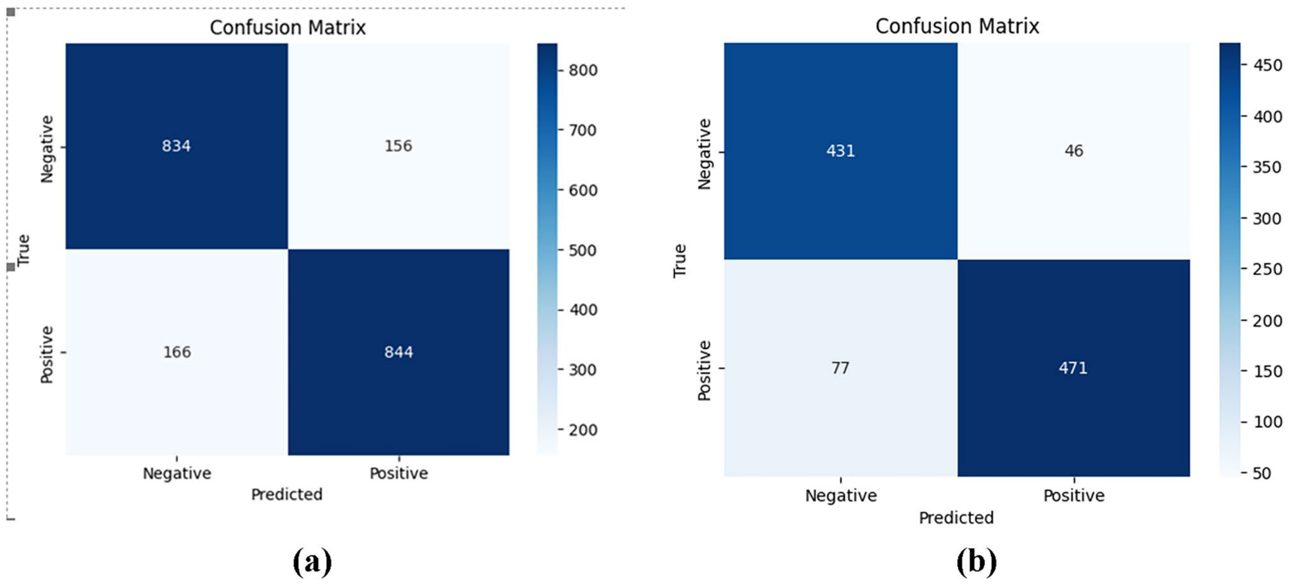

Fig. 7. Confusion matrix for hypergraph (**a**) and HGATT_LR (**b**) model.

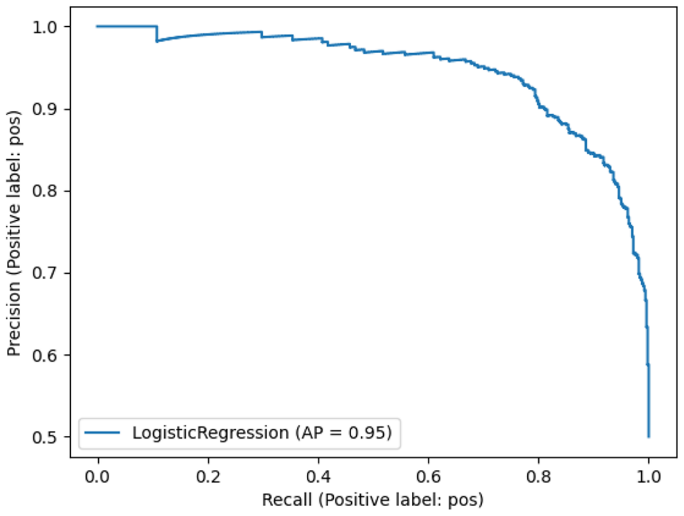

Fig. 8. Precision-recall curveHGATT_LR model.

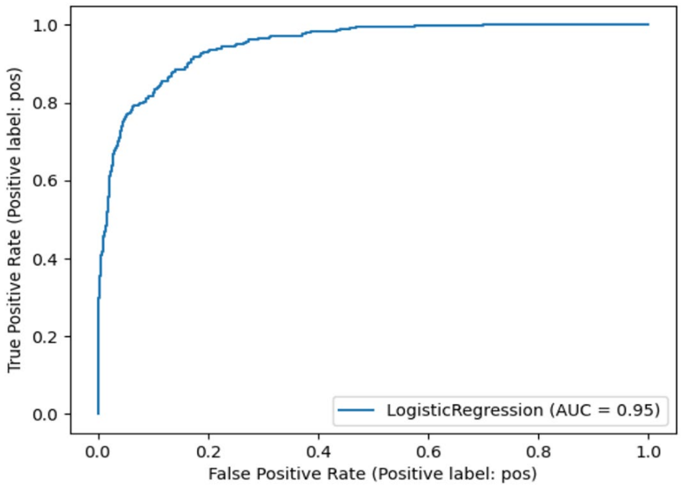

Fig. 9. ROC curve for HGATT_LR model.

Model used in our	Accuracy obtained in our model	Precision obtained	Recall obtained	F1-score
Hypergraph	84.38	83.42	83.40	83.40
HGATT_LR model	88.00	87.00	87.00	88.00

Table 6. The record of parameters for Hypergraph model.

Data manipulation, visualization, NLP, and ML are all done using libraries. Dataset is loaded into a Pandas data frame, and any rows with blank or missing values are removed. NLTK's Vader emotion intensity analyzer is used to assign each review a sentiment score. A confusion matrix, precision-recall curve, and ROC curve are used to evaluate the sentiment analysis's performance. On the training set and the testing set, a logistic regression model is calibrated.

According to the above results, the Hypergraph-Logistic Regression model has an accuracy of 88 which is the highest compared to other models in the Table 7. Further evidence that the hypergraph model is superior can be seen in the fact that the level of accuracy, recall, and the F1 score for the hypergraph-logistic regression model is much greater than those for the latter.

Model	Accuracy
Decision tree	80.45
Naïve Bayer	82.61
Multinomial NB	82.04
Sentiment intensity analyzer	70.07
Gaussian NB	83.21
HGATT_LR (proposed)	88.00

Table 7. The record of accuracy of different models.

Conclusion

This study presents a novel model for predicting consumer sentiment using the Hypergraph attention mechanism. The model outperforms the baseline model in terms of accuracy, precision, recall, and F1-score, demonstrating its effectiveness in forecasting consumer reviews. However, the existing approach has limitations, such as a limited set of features and computational efficiency. This could lead to inferior performance in complex or diverse text input, especially when dealing with different domains or languages. The model's scalability and computational efficiency may be constrained when processing large amounts of text data, resulting in longer training times or resource-intensive computations. One significant advantage of the suggested approach is its scalability, since it can easily accommodate additional training data to improve its performance. Because of its scalability, it is an invaluable tool for product managers seeking to understand their customers' wants and preferences. Product managers can acquire insights into client attitudes and make informed decisions to design goods that meet customer expectations by analyzing consumer reviews. The proposed approach extends its applicability beyond sentiment analysis. It can be employed in various other domains, such as opinion mining and customer feedback analysis, to extract valuable insights from textual data. By leveraging the power of machine learning and natural language processing, the proposed method offers a versatile solution for understanding and analyzing customer sentiments. In conclusion, the proposed machine learning-based approach, with its utilization of the Hypergraph attention mechanism, presents a robust solution for predicting consumer sentiment. Its superior performance, scalability, and applicability to various domains make it a valuable tool for product managers and researchers in understanding and utilizing consumer feedback effectively.

Data availability

The data that support the findings of this study are publicly available online at Kaggle.https://www.kaggle.com/code/mammadabbasli/amazon-reviews-analysis-logisticregression/input

Received: 1 May 2024; Accepted: 19 August 2024
Published online: 23 August 2024

References

1. Y. Wang, M. Huang, X. Zhu, & L. Zhao. Attention-based LSTM for aspect-level sentiment classification. in *EMNLP* (2016).
2. S. I. Wang & C. D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2012), pp. 90–94.
3. Miotto, R., Li, L., Kidd, B. A. & Dudley, J. T. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Sci. Rep.* **6**(1), 1–10 (2016).
4. Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint* arXiv:1408.5882 *(2014).*
5. P. Liu, X. Qiu, & X. Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint* arXiv:1605.05101 *(2016).*
6. Yao, L., Mao, C. & Luo, Y. Graph convolutional networks for text classification. *Proc. AAAI Conf. Artif. Intelligence* **33**(01), 7370–7377 (2019).
7. K. Ding, J. Wang, J. Li, D. Li, & H. Liu. Be more with less: Hypergraph attention networks for inductive text classification. *arXiv preprint* arXiv:2011.00387 *(2020).*
8. R. P. S. Joseph. Amazon reviews sentiment analysis: A reinforcement learning approach. Doctoral dissertation, MS Thesis, Griffith College Dublin, Ireland (2020).
9. R. Gandhi. Sentiment analysis through LSTMs. (2018).
10. Y. Zhang, M. Guo, Q. Yan, & G. Shen. Short text classification via hypergraph convolution network. in *2021 3rd International Symposium on Smart and Healthy Cities (ISHC)*, pp. 72–76, IEEE (2021).
11. A. Kolawole John, L. Di Caro, L. Robaldo, & G. Boella. Textual inference with tree-structured LSTM. in *BNAIC 2016: Artificial Intelligence: 28th Benelux Conference on Artificial Intelligence, Amsterdam, The Netherlands, November 10–11, 2016, Revised Selected Papers 28*, pp. 17–31 (Springer International Publishing, 2017).
12. S. K. Sønderby, C. K. Sønderby, H. Nielsen, & O. Winther. Convolutional LSTM networks for subcellular localization of proteins. in *Algorithms for Computational Biology: Second International Conference, AlCoB 2015, Mexico City, Mexico, August 4–5, 2015, Proceedings 2*, pp. 68-80 (Springer International Publishing, 2015).
13. J. Wang, L. C. Yu, K. R. Lai, & X. Zhang. Dimensional sentiment analysis using a regional CNN-LSTM model. in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 225–230 (2016).
14. Wang, J., Yu, L. C., Lai, K. R. & Zhang, X. Tree-structured regional CNN-LSTM model for dimensional sentiment analysis. *IEEE/ACM Trans. Audio Speech Lang. Process.* **28**, 581–591 (2019).
15. P. Juyal. Sentimental analysis of amazon customers based on their review comments. in *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, pp. 914–919, IEEE (2022).
16. X. Shi & R. Lu. Attention-based bidirectional hierarchical LSTM networks for text semantic classification. in *2019 10th International Conference on Information Technology in Medicine and Education (ITME)*, pp. 618–622, IEEE (2019).

17. B. K. Shah, A. K. Jaiswal, A. Shroff, A. K. Dixit, O. N. Kushwaha, & N. K. Shah. Sentiments detection for Amazon product review. in *2021 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–6, IEEE (2021).
18. S. Bird. NLTK: The natural language toolkit. in *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 69–72 (2006).
19. Zhang, W., Yoshida, T. & Tang, X. A comparative study of TF* IDF, LSI and multi-words for text classification. *Expert Syst. Appl.* **38**(3), 2758–2765 (2011).
20. R. Rawat, V. Mahor, S. Chirgaiya, R. N. Shaw, & A. Ghosh. Analysis of darknet traffic for criminal activities detection using TF-IDF and light gradient boosted machine learning algorithm. in *Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2021*, pp. 671–681 (Springer Singapore, 2021).
21. Blei, D. M., Ng, A. Y. & Jordan, M. I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003).
22. Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, & E. Hovy. Hierarchical attention networks for document classification. in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489 (2016).
23. Bai, S., Zhang, F. & Torr, P. H. Hypergraph convolution and hypergraph attention. *Pattern Recognit.* **110**, 107637 (2021).
24. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014).
25. A. F. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint* arXiv:1803.08375 *(2018).*
26. Liu, G. & Guo, J. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing* **337**, 325–338 (2019).
27. Yang, Y. An evaluation of statistical approaches to text categorization. *J. Information Retrieval* **1**(1/2), 67–88 (1999).
28. Ikonomakis, M., Kotsiantis, S. & Tampakas, V. Text classification using machine learning techniques. *WSEAS Trans. Computers* **4**(8), 966–974 (2005).
29. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning* 2nd edn. (Springer Verlag, 2009).
30. Maalouf, M. Logistic regression in data analysis: An overview. *Int. J. Data Anal. Tech. Strategies* **3**(3), 281–299 (2011).
31. Hosmer, D. W. & Lemeshow, S. *Applied Logistic Regression* 2nd edn. (Wiley, 2000).
32. Heydarian, M., Doyle, T. E. & Samavi, R. MLCM: Multi-label confusion matrix. *IEEE Access* **10**, 19083–19095 (2022).

Acknowledgements
This study was supported by the Research Program funded by the Seoul National University of Science and Technology (SeoulTech).

Author contributions
P.S., E.J. V.S. and M.H. contributed to the writing of the main manuscript text. G.D., A.K. and D.K. provided the foundational ideas for the research. D.K. also provided financial support for the project. All authors critically reviewed and approved the final manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to D.K.

Reprints and permissions information is available at www.nature.com/reprints.