COMP4432 Machine Learning

Data Product: A Million News Headlines

Group Project Report

Group member:

ZHENG Shouwen 21106717D

ZHOU Taiqi 21097982D

Feng Yuqi 20054248D

Comprehensive news analysis framework based on the data set "A Million News Headlines"

Abstract

The "A Million News Headlines" dataset, obtained from Kaggle, contains a comprehensive collection of approximately one million news headlines published over a period of nineteen years by the Australian news source ABC (Australian Broadcasting Corporation).

The project focuses on formulating, designing, and implementing a machine learning solution to cluster the headlines, perform topic modeling using Latent Semantic Analysis (LSA), and classify the headlines. Additionally, an extended study is conducted to classify fake news headlines. The performance of the implemented methods is evaluated, and improvements are proposed. The findings and analyses are reported in this project, which contributes to understanding the underlying patterns and characteristics of the news headlines.

1. Introduction

1.1 Project background and process

News headlines are essential in modern information-rich world since they offer a succinct synopsis of news articles, grab readers' attention, and influence how they interpret current events. The exponential proliferation of news items makes it difficult to analyze and arrange massive amounts of headlines. The goal of this project is to apply machine learning techniques to take the "A Million News Headlines" dataset and turn it into relevant insights.

The dataset contains a sizable collection of headlines from nineteen years of news. Through the use of this dataset, we intend to achieve a number of important goals. In order to find trends and put related headlines together, we will first examine how the news headlines are clustered.

Next, we will conduct topic modeling on the headlines using Latent Semantic Analysis (LDA). We can identify the underlying themes and content clusters in the news headlines by extracting latent topics from the dataset. This analysis will provide a deeper understanding of the prevalent topics and trends discussed over the years.

Then we will delve into headline classification, where we aim to develop a machine learning model that can accurately categorize headlines into different predefined classes. This classification task will enable us to automatically assign

labels to new headlines based on their content, making it easier to organize and navigate through vast amounts of news information.

In addition to clustering, topic modeling, and classification, we will conduct sentiment analysis on the news headlines. Sentiment analysis involves determining the emotional tone or sentiment expressed in the text. By applying sentiment analysis techniques, we can gauge the overall sentiment (positive, negative, or neutral) associated with each headline. This analysis will provide insights into the emotional landscape of the news over time and help identify patterns or trends in public sentiment.

Furthermore, we will explore the concept of news hotness modeling. Hotness refers to the level of attention for a particular news topic receives at a given time. By analyzing the frequency and distribution of headlines related to specific topics, we can develop a model to quantify and predict the hotness of news stories. This hotness modeling will enable us to identify trending topics, anticipate public interest, and understand the dynamics of news cycles.

Apart from these key objectives, we will broaden the scope of our investigation to address the categorization of fake news headlines. The proliferation of propaganda and false information makes it imperative to create techniques for separating reputable sources from fake news. To help with the ongoing attempts to stop fake news from spreading, we want to train a classifier on tagged false and real news headlines.

Through this project, we seek to gain insights into the characteristics and patterns of news headlines, improve the machine learning methods employed, and provide a comprehensive analysis and evaluation of our findings. The results obtained will contribute to the field of natural language processing and further our understanding of news headline analysis and classification.

1.2 Data Preprocessing and Exploratory Data Analysis

Before applying various machine learning algorithms to explore the latent semantic space of the provided dataset, we conducted several data preprocessing steps to clean and prepare the data for analysis. The preprocessing steps included:

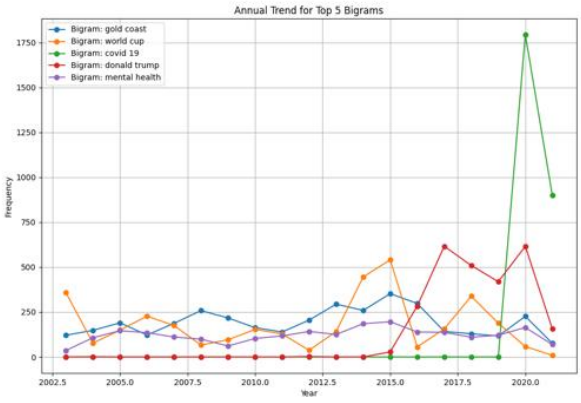
· Text Preprocessing: The headline text was converted to lowercase, tokenized into individual words, and filtered to remove stopwords using the NLTK library. Additionally, stemming was performed using the Porter Stemmer to reduce words to their base or root form.

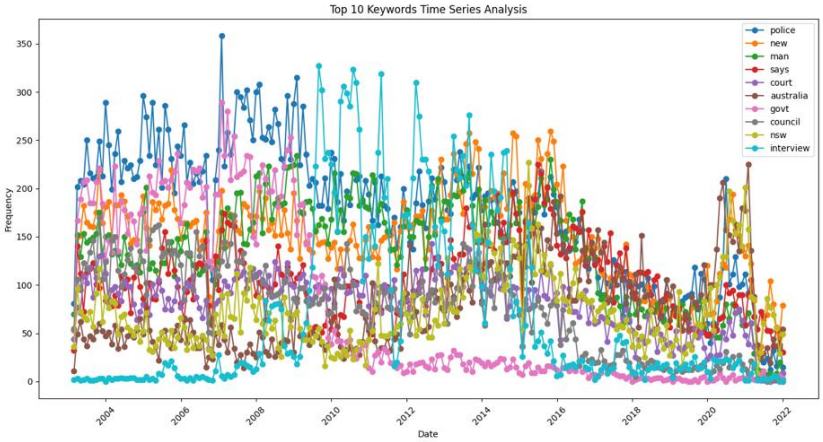
· Date Feature Extraction: The 'publish_date' column was converted to datetime format, and new features such as 'year', 'month', and 'day' were extracted from the date. These features were then scaled using the StandardScaler from the sklearn library.

· Data Cleaning: Duplicate headlines were removed from the dataset to ensure data integrity and avoid bias in subsequent analyses.

After the preprocessing steps, the dataset was saved as "processed_data.csv" for further analysis and modeling. Besides, Some basic exploratory data analysis was performed to gain initial insights into the dataset. The analysis revealed that the total number of headlines in the dataset is 1,244,185, with 1,213,004 unique words after preprocessing. The publish dates range from February 19, 2003, to December 31, 2021.

Of course, we also used some statistical algorithms to make simple estimates of the dataset. Out of the 1244185 data, we extracted some keywords and phrases with the highest frequency of occurrence





(Figures1&2. Top keywords and phrases)

2. Fake News Classification

2.1 Data cleaning and preprocessing

During this process, we conducted a series of detailed data cleaning and pre-processing steps to ensure that the data set used for the true and false news identification model was of high quality and consistent.

First, load the real news and fake news data from a CSV file. To ensure data integrity, we removed rows containing any missing values and removed all duplicate news headlines from the dataset to increase the accuracy of the data. In addition, the data column names were unified and the title_text column was renamed to ensure that the column names of the data set are consistent during subsequent processing.

For more efficient text analysis, we integrated English stopword lists from three libraries: NLTK, Gensim, and Scikit-learn to create a comprehensive stopword collection. To be able to remove words in the preprocessing stage that usually do not carry useful information in text analysis.

Next, the news headlines underwent thorough text preprocessing, including converting all text to lowercase to unify capitalization, removing all non-alphabetic characters to ensure text purity, and stemming to simplify vocabulary, which helps improve follow-up Processing efficiency. Finally, stop words are removed from the text to clean and prepare the data for model training.

When building a classification supervised learning model, all real news is labeled fake = 0 and fake news is labeled fake = 1. In addition, in order to balance the proportion of real news and fake news in the data set, we selected the top 50,000 real news and 20,000 fake news in the respective data sets. This step is crucial as it helps avoid sample bias during training.

Finally, we divided the data set into a training set and a test set in a ratio of 4:1. On this basis, training and testing data sets are constructed by adding random noise and conducting random sampling to ensure the representativeness of the data and the generalization ability of the model.

2.2 Feature construction

During the feature construction process, we use the TF-IDF (term frequency-inverse document frequency) vectorization method to convert the text data of news titles into numerical features, which is crucial for subsequent model training.

In this process, we create a TF-IDF vectorizer and set the maximum number of features to 1000. This means that the vectorizer only considers the most important 1000 words in the entire dataset. In this way, we are able to improve the efficiency of model training while reducing the potential negative impact of less informative features on model performance. This strategy helps ensure that the feature set contains only the most representative and discriminative words, thus optimizing the model training process and final classification performance.

2.3 Traditional machine learning model training

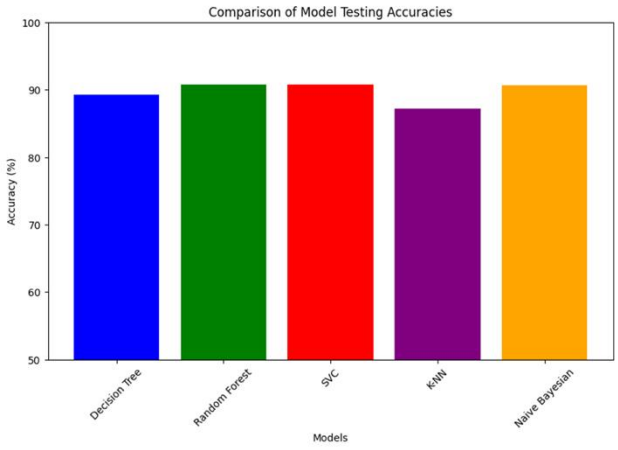
In this study, we adopted a variety of traditional machine learning models for news classification tasks, including decision tree classifiers, random forest classifiers, support vector machines, K nearest neighbors, and polynomial naive Bayes. Each model has its specific application scenarios and advantages:

1) Decision Tree Classifier: This model is suitable for processing non-linear data. It has the advantage that the model results are easy to understand and interpret, and is very suitable for preliminary analysis of the structure of the data.
2) Random Forest Classifier (RandomForestClassifier): As a method based on ensemble learning, random forest usually shows high accuracy and stability, and can effectively handle complex data sets with a large number of features.
3) Support Vector Machine (SVC): We use SVC with a linear kernel. This method is particularly suitable for data sets with a large number of features, such as text classification tasks, and it can effectively find decision boundaries in high-dimensional space.
4) K Nearest Neighbors (KNeighborsClassifier): This is an instance-based learning method best suited for small data sets. However, its performance may suffer when feature dimensions are higher because distance measures may become less effective in high-dimensional spaces.
5) Multinomial Naive Bayes (MultinomialNB): This model is particularly suitable for processing count or frequency data and is often used for the classification of text data because it assumes that the probability distribution of feature-generated data obeys the multinomial distribution.

The following figures are the accuracy of each models:

```
------------------------------
Testing Accuracy of Decision Tree: 89.26 %
------------------------------
Testing Accuracy of Random Forest: 90.79 %
------------------------------
Testing Accuracy of SVC: 90.8 %
------------------------------
Testing Accuracy of K-NN: 87.19 %
------------------------------
Testing Accuracy of Naive Bayesian: 90.68 %
------------------------------
```

(Figures2. The accuracy of the running models in code line)



(Figures3. Column comparison of each model's accuracy)

Preliminary model training results show that the random forest classifier performs best, showing good classification accuracy and robustness. Based on this finding, we will further refine and tune the random forest model

2.4 Random forest model optimization

To optimize the performance of the random forest classifier, we implemented a systematic parameter tuning process using grid search coupled with three-fold cross-validation to identify model parameters that best fit our data. In grid search, we define ranges for several parameters, including the number of trees, the maximum depth of trees, the minimum number of samples required for node splitting, and the number of features considered. This process aims to find the optimal model configuration by systematically exploring different parameter combinations and evaluating the performance of each combination under a cross-validation framework.

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Best parameters found:  {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 157}
Best accuracy found:  0.9115965063556566
[CV] END max_depth=10, min_samples_split=4, n_estimators=174; total time=   2.3s
[CV] END max_depth=20, min_samples_split=9, n_estimators=216; total time=   4.6s
[CV] END max_depth=10, min_samples_split=6, n_estimators=188; total time=   2.5s
[CV] END max_depth=20, min_samples_split=5, n_estimators=192; total time=   4.1s
[CV] END max_depth=10, min_samples_split=9, n_estimators=257; total time=   3.5s
[CV] END max_depth=10, min_samples_split=6, n_estimators=188; total time=   2.5s
[CV] END max_depth=10, min_samples_split=4, n_estimators=174; total time=   2.1s
[CV] END max_depth=20, min_samples_split=9, n_estimators=216; total time=   4.4s
[CV] END max_depth=10, min_samples_split=3, n_estimators=291; total time=   3.6s
[CV] END max_depth=20, min_samples_split=5, n_estimators=192; total time=   4.1s
[CV] END max_depth=10, min_samples_split=9, n_estimators=257; total time=   3.5s
[CV] END max_depth=10, min_samples_split=6, n_estimators=188; total time=   2.5s
[CV] END max_depth=20, min_samples_split=5, n_estimators=192; total time=   4.1s
[CV] END max_depth=20, min_samples_split=7, n_estimators=152; total time=   3.2s
[CV] END max_depth=10, min_samples_split=3, n_estimators=291; total time=   3.8s
[CV] END max_depth=10, min_samples_split=4, n_estimators=174; total time=   2.3s
[CV] END max_depth=20, min_samples_split=9, n_estimators=216; total time=   4.6s
[CV] END max_depth=10, min_samples_split=3, n_estimators=291; total time=   3.8s
[CV] END max_depth=None, min_samples_split=6, n_estimators=202; total time= 1.1min
[CV] END max_depth=None, min_samples_split=6, n_estimators=202; total time= 1.2min
[CV] END max_depth=None, min_samples_split=6, n_estimators=202; total time= 1.2min
[CV] END max_depth=20, min_samples_split=4, n_estimators=171; total time=   3.7s
[CV] END max_depth=10, min_samples_split=9, n_estimators=257; total time=   3.3s
[CV] END max_depth=None, min_samples_split=2, n_estimators=157; total time= 1.1min
[CV] END max_depth=20, min_samples_split=4, n_estimators=171; total time=   3.7s
[CV] END max_depth=20, min_samples_split=7, n_estimators=152; total time=   3.2s
[CV] END max_depth=None, min_samples_split=2, n_estimators=157; total time= 1.1min
[CV] END max_depth=20, min_samples_split=4, n_estimators=171; total time=   3.7s
[CV] END max_depth=20, min_samples_split=7, n_estimators=152; total time=   3.2s
[CV] END max_depth=None, min_samples_split=2, n_estimators=157; total time= 1.1min
```

(Figures4. Running code line of the RF Optimization)

2.5 Deep machine learning model training

2.5.1 DNN model

- Model architecture:

Build a neural network using the Sequential model in the Keras library. This model allows us to stack individual layers in a linear fashion.

- Network layer configuration:

The first layer is a fully connected layer (Dense), configured with 128 neurons, and uses the ReLU activation function to increase the nonlinear processing capability of the network.

Dropout layer: Immediately after the Dense layer, set to 0.2, that is, 20% of the activated neurons are randomly set to zero during the training process. This helps prevent the model from overfitting and improves its generalization on new data. performance.

Another layer of Dense and Dropout layers: added with the same configuration to enhance the expression ability of the model.

Output layer: Finally, a Dense layer with a single neuron is used. The activation function is sigmoid. It is specially designed to handle binary classification problems and output the probability of predicting a positive class.

- Compile the model:

Loss function: Binary cross entropy (binary_crossentropy) is used, which is suitable for dealing with binary classification problems.

Optimizer: Choose Adam, which can automatically adjust the learning rate to help better optimize the model under various network conditions.

Performance Metrics: Set to Accuracy to visually evaluate the model's performance on training and validation data.

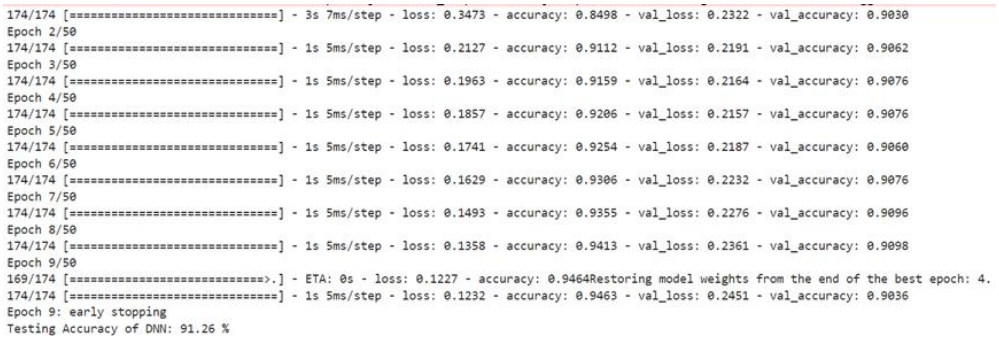
Early Stopping: In order to avoid overfitting during the training process and save training resources, the early stopping callback function is set to stop training when the verification loss does not improve within 5 consecutive cycles and restore to the best performance. model status.

- Training configuration

Validation set: 10% is divided from the training data to periodically verify the model performance to ensure that the model has good adaptability to unseen data.

Training period (epochs): Set to 50 iterations, balancing training depth and computing efficiency.

Batch size: Set to 256, this helps maintain stability with each weight update and speeds up the training process.



(Figures5. Running code line of the RNN training)

2.5.2 CNN model

- One-dimensional convolutional layer (Conv1D):

As the first layer of the model, 64 filters are used, and the kernel size of each filter is set to 3. This allows the model to capture local features in text data and is suitable for processing serialized text data.

ReLU is selected as the activation function to increase the nonlinear processing capability of the network.

Explicitly specify the input shape as (tfidf_train_dense.shape[1], 1), where tfidf_train_dense.shape[1] represents the number of features and 1 represents the number of channels.

- Max pooling layer (MaxPooling1D):

The first pooling uses a window size of 2, which helps reduce the computational effort while retaining important feature information.

The second time uses a similar configuration, but the number of filters in the convolutional layer is increased to 128 to enhance the model's ability to capture more complex features.

- Global maximum pooling layer (GlobalMaxPooling1D):

Extract the most salient features from all convolution outputs, ensuring that the dimensions of the output features are independent of the size of the input.

- Fully connected layer (Dense):

Two dense layers are next added to the network, the first with 256 nodes and the second with 128 nodes for further processing of the features extracted from the convolutional and pooling layers.

Each Dense layer is followed by a Dropout layer, and its ratio is set to 0.3, that is, 30% of the activation units are randomly set to zero during the training process, effectively reducing model overfitting.

- Output layer:

Finally, a Dense layer with a single node and a sigmoid activation function are used to output the probability of belonging to the positive class, which is suitable for handling binary classification problems.

- Loss functions and optimization

Loss function: Choose binary cross entropy (binary_crossentropy), which is the standard loss function for binary classification problems.

- Compilation and training:

Use the Adam optimizer, which can dynamically adjust the learning rate to help the model learn at an appropriate rate during different training stages.

Early stopping technology is also used during model training, with verification loss as the monitoring indicator. If there is no improvement for five consecutive cycles, the training will be stopped and rolled back to the best model state.

```
174/174 [==============================] - 2s 10ms/step - loss: 0.4563 - accuracy: 0.8184 - val_loss: 0.4613 - val_accuracy: 0.8124
Epoch 25/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4563 - accuracy: 0.8177 - val_loss: 0.4649 - val_accuracy: 0.8104
Epoch 26/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4558 - accuracy: 0.8187 - val_loss: 0.4689 - val_accuracy: 0.8092
Epoch 27/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4549 - accuracy: 0.8192 - val_loss: 0.4637 - val_accuracy: 0.8092
Epoch 28/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4565 - accuracy: 0.8177 - val_loss: 0.4611 - val_accuracy: 0.8126
Epoch 29/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4549 - accuracy: 0.8192 - val_loss: 0.4614 - val_accuracy: 0.8114
Epoch 30/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4552 - accuracy: 0.8187 - val_loss: 0.4647 - val_accuracy: 0.8098
Epoch 31/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4547 - accuracy: 0.8185 - val_loss: 0.4625 - val_accuracy: 0.8112
Epoch 32/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4544 - accuracy: 0.8196 - val_loss: 0.4609 - val_accuracy: 0.8122
Epoch 33/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4547 - accuracy: 0.8187 - val_loss: 0.4615 - val_accuracy: 0.8124
Epoch 34/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4546 - accuracy: 0.8191 - val_loss: 0.4616 - val_accuracy: 0.8124
Epoch 35/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4549 - accuracy: 0.8184 - val_loss: 0.4605 - val_accuracy: 0.8124
Epoch 36/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4541 - accuracy: 0.8183 - val_loss: 0.4608 - val_accuracy: 0.8120
Epoch 37/50
174/174 [==============================] - 2s 11ms/step - loss: 0.4548 - accuracy: 0.8189 - val_loss: 0.4614 - val_accuracy: 0.8122
Epoch 38/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4548 - accuracy: 0.8196 - val_loss: 0.4606 - val_accuracy: 0.8130
Epoch 39/50
174/174 [==============================] - 2s 10ms/step - loss: 0.4542 - accuracy: 0.8186 - val_loss: 0.4626 - val_accuracy: 0.8096
Epoch 40/50
170/174 [============================>.] - ETA: 0s - loss: 0.4542 - accuracy: 0.8191Restoring model weights from the end of the best epoch: 35.
174/174 [==============================] - 2s 10ms/step - loss: 0.4539 - accuracy: 0.8191 - val_loss: 0.4619 - val_accuracy: 0.8130
Epoch 40: early stopping
```

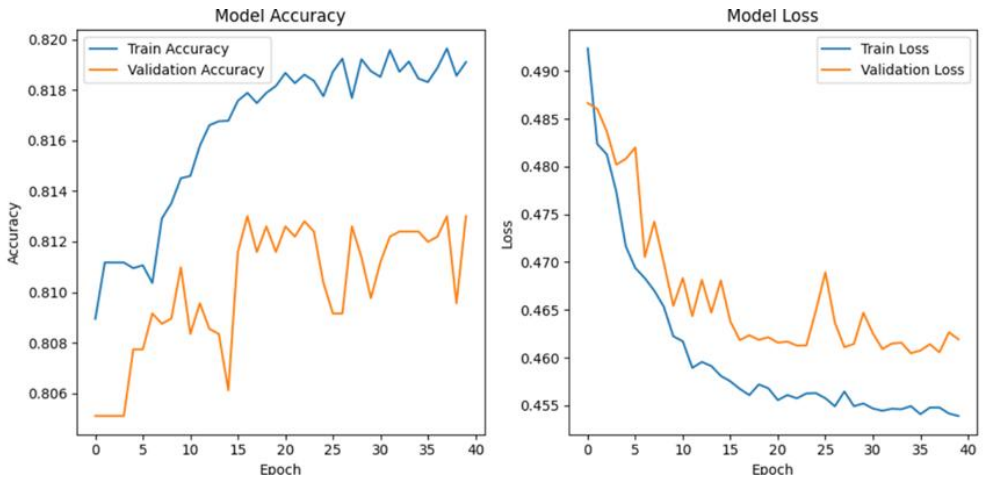
(Figures6. Running code line of the CNN training)

2.6 Training and validation metric analysis

During the training process of the model, monitoring of training and validation metrics is crucial, as they can provide us with valuable information about model performance and learning progress. In particular, by plotting the accuracy and loss curves of the model during training and validation, we can deeply analyze the behavior of the model and better understand its performance at each stage.

These figures show the performance changes of the model in each training cycle (Epoch), helping us identify whether the model has overfitting or underfitting problems:

- Overfitting usually manifests itself as a continuous decrease in training loss, while the validation loss starts to increase after a certain point.
- Underfitting is characterized by high training loss and verification loss that decrease slowly.



(Figures7. Model accuracy and loss change with each epoch)

2.7 Result evaluation

In this progress, we use both traditional machine learning methods and deep machine learning technology to train models to handle specific classification tasks. By comprehensively comparing the performance of different models on the training and validation sets, we identify the models that perform best in their respective technical categories. The comparison table below details the performance indicators of various models:

Models		Accuracy
Tradition al ML	DT	89.26 %
	RF	90.79 %
	SVC	91.16 %
	KNN	87.19 %
	NB	90.68 %
Deep ML	DNN	90.99 %
	CNN	81.96%

(Table. Model accuracy Table)

3. Clustering Analysis

3.1 Data cleaning and preprocessing

- Text preprocessing

SnowballStemmer: Use this stemmer to restore words to their base form (stem), helping to reduce the complexity of inflection.

RegexpTokenizer: Apply regular expressions to match alphabetic sequences and exclude all non-alphabetic characters (such as punctuation) to clean text data.

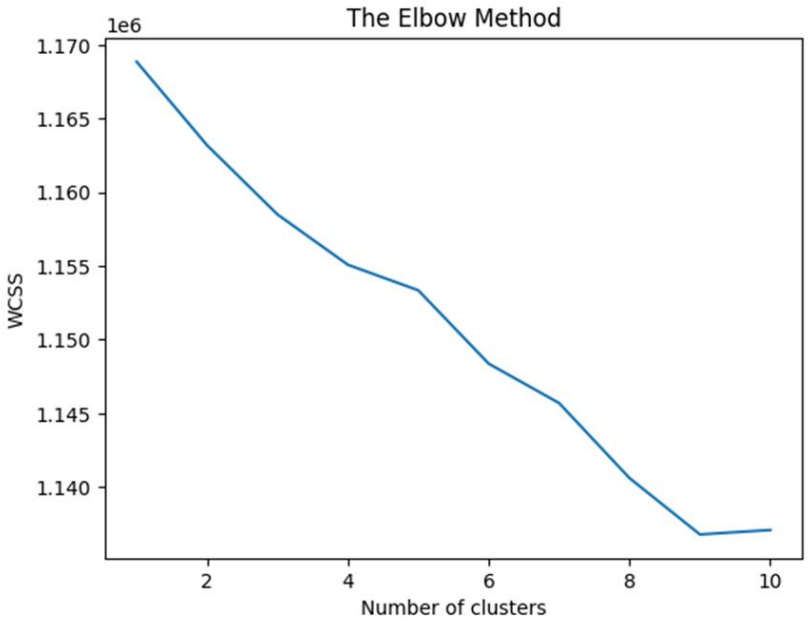
Custom word segmentation processing: configure the word segmenter, process the text into lowercase, unify the data format, and obtain the stem of each word through the stem extractor to prepare for subsequent TF-IDF vectorization.

- Feature extraction

TF-IDF vectorization: Use TfidfVectorizer to convert the processed text data into TF-IDF format, which can effectively reflect the importance of words in the document collection. The vectorizer is configured with stop word lists and custom word segmentation and stemming capabilities. Additionally, setting max_features=1000 limits the number of features the model considers, ensuring the model focuses on the most critical words in the data.

- Determination of the number of clusters

Elbow's rule: To determine the optimal number of clusters, we use the K-means algorithm to test different numbers of clusters in the range of 1 to 10 and calculate the WCSS (within-cluster sum of squares) in each case. WCSS is a key indicator of clustering effect, which measures the sum of squared distances from each point to its cluster center. Observing the decreasing trend of WCSS value as the number of clusters increases, it is usually found that the decrease of WCSS value slows down significantly at a certain point. This point is often considered the "elbow point" and represents the optimal number of clusters.



(Figures8. Elbow plot: The x-axis represents the number of clusters, and the y-axis represents the corresponding WCSS value. By looking for the "elbow" in the figure, the optimal number of clusters can be roughly determined.)

Select the best k value position, divide the clusters, and print out the corresponding clustering feature words. Then the clustering results will be saved as a label container and stored in a new csv for subsequent sentiment analysis and topics. Modeling.

3.2 Deep learning model building part

(i) Bert model based on transformer

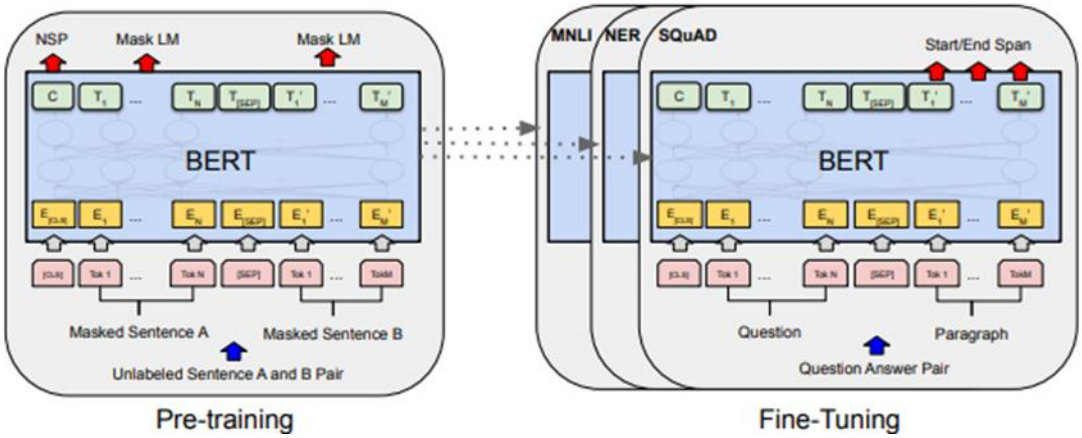
In this process, we use the pre-trained deep learning model BERT (Bi Direction Encoder Representations from Transformers) based on the Transformers architecture to process text data. The BERT model is built on the attention mechanism and is specifically designed to capture bidirectional contextual relationships in text, which is its main difference from traditional one-way or separate bi-directional language models. By comprehensively considering the contextual information on the left and right sides of each layer, BERT can understand the text content more deeply.

BERT's pre-training includes two main tasks:

Masked Language Model (MLM): In this task, the model randomly masks certain words in the text (using the [mask] tag instead) and then tries to predict these masked words. This process enables BERT to effectively learn rich language features and dependencies between words.

Next Sentence Prediction (NSP): In this task, the model predicts whether two sentences are logically coherent. This helps BERT learn relationships between sentences, which is crucial for understanding document structure and sentence coherence.

In the project, we first load the pre-trained BERT model and its related components from the HuggingFace platform. With appropriate configuration, we can tailor the model to meet specific application needs. Leveraging the BERT model, we extract deep semantic embeddings from text data, capturing the fine-grained features and contextual relationships of the text.



（Figures9. Bert Model Framework）

Clustering and label generation

The resulting text embeddings are then used in K-means cluster analysis to explore and identify potential patterns and groups in the data. Through this cluster analysis, we generate corresponding tags for each piece of text, which can subsequently be used for advanced text analysis tasks such as sentiment analysis and topic modeling.

(ii) Autoencoder

Not only that, we also tried the autoencoder framework, which allows us to transform the dimensions of feature vectors up and down for unsupervised clustering learning. The model is divided into the following parts:

- Encoder:

Responsible for converting high-dimensional input data into low-dimensional hidden representation (encoding). This process is implemented step by step through

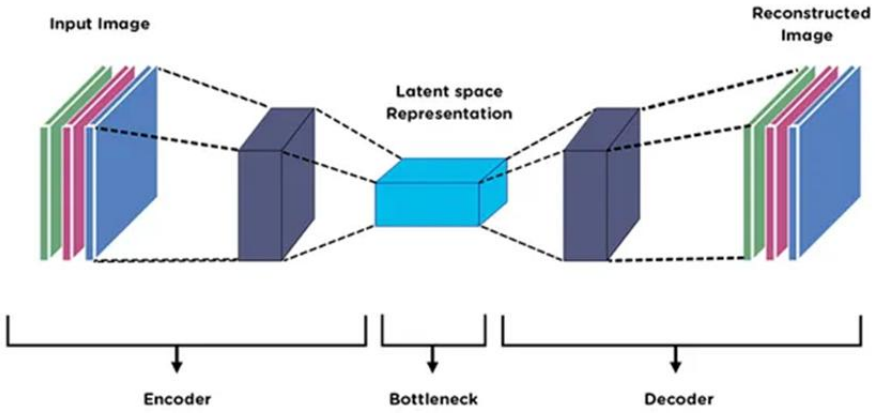
multiple layers, each layer reducing the dimensionality of the data until a set compression level is reached.

- Bottleneck layer:

At the end of the encoding process, a compressed representation of the data is output. The dimensionality of this layer is much smaller than the input layer, forcing the model to learn a refined and compressed representation of the data, which is key to capturing the main features of the input data.

- decoder:

Responsible for reconstructing the original input data from the compressed representation of the bottleneck layer. The decoding process typically involves multiple incremental layers that gradually expand the data dimensions back to the size of the original input.



(Figures10. Bert Model Framework)

In order to adapt to the input requirements of the Keras model, we first convert the sparse matrix generated by the TF-IDF algorithm into a dense matrix format. During the model training phase, early stopping technology (EarlyStopping) is used to monitor the loss on the validation set to avoid overfitting and terminate training early when the loss no longer improves. This strategy not only ensures the generalization ability of the model, but also improves training efficiency.

After completing the training, we used the encoding part of the autoencoder to extract features from all the data, and these low-dimensional feature representations were subsequently used for K-means cluster analysis. We analytically determined the most appropriate number of clusters and delved into the characteristics and

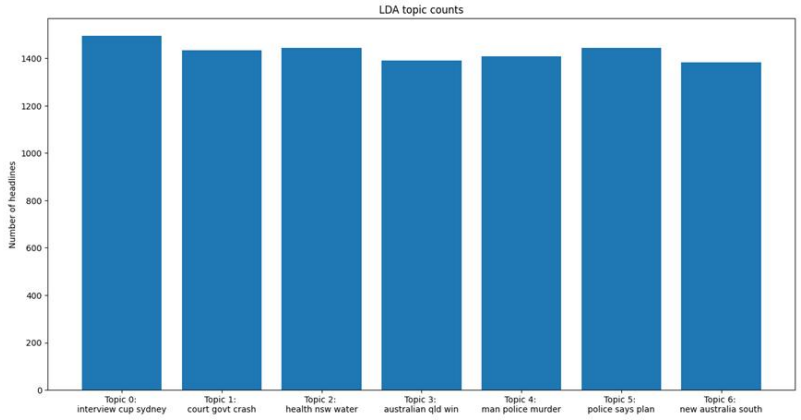
similarities of the data within each cluster, a critical step in understanding and leveraging the underlying structure of the data.

4. LDA Topic Modeling

In this part, we first preprocess and vectorize the text data to convert it into a term frequency (TF) matrix. This is a commonly used text representation method that quantifies the frequency of words in each document and provides a suitable input format for machine learning algorithms.

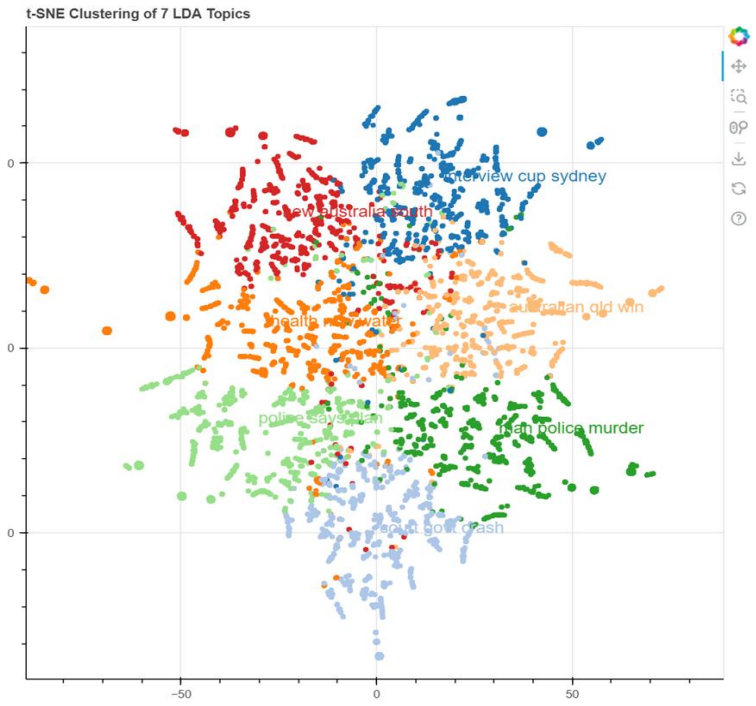
- Topic modeling and visualization

Topic Modeling: We apply the Latent Dirichlet Allocation (LDA) model for topic modeling. LDA is a widely used statistical model that assumes that each document consists of multiple topics, and each topic consists of a sequence of words with high probability. Through LDA, we extract multiple topics from text data, each topic is characterized by words frequently associated with it, revealing the underlying semantic structure of the document.



(Figure 11. Extract the top three words of the topic and plot the topic count chart and then plot the number of documents under each topic. Understand the main content of each topic and their prevalence in the data set)

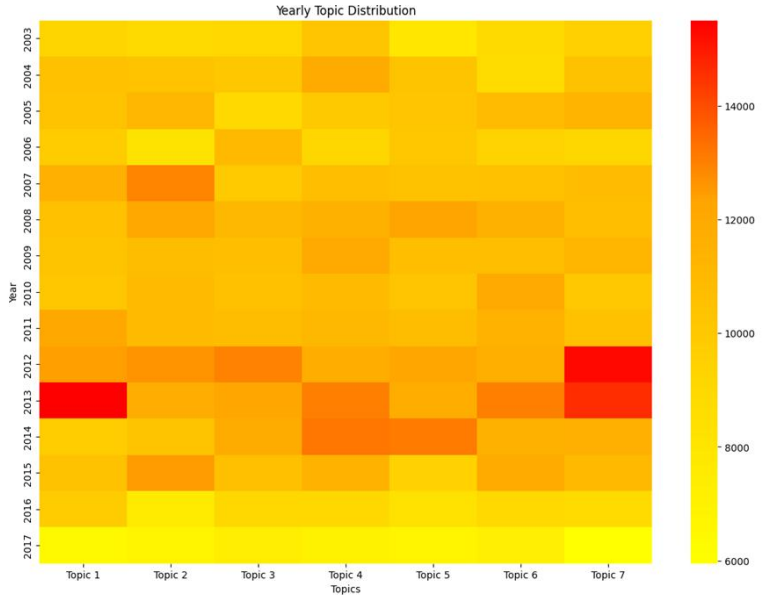
Data visualization: Use t-SNE technology to reduce the dimensionality of the high-dimensional topic model output and map it to a two-dimensional space. This nonlinear dimensionality reduction technology helps us intuitively cluster documents with similar topics and separate documents with different topics, thereby visually displaying the similarities and differences between documents.



*(*Figures *12: The t-SNE algorithm reduces the dimensionality of the topics generated by LDA to a two-dimensional space to visualize the relative positions and distributions between topics. To observe whether the topics identified by the model have obvious distinction.)*

- Long term topic tracking

In addition, we conducted a year-by-year thematic analysis of news headlines from 2003 to 2017. By applying LDA to each year's data, we are able to track and analyze the evolution of each topic over time. These annual topic data are aggregated and used for time series analysis using heat maps to show the popularity of different topics each year. This not only reveals changes in the focus of public attention over time, but also reflects the evolution of socio-cultural trends.



*(*Figures13. *Shows the distribution of each topic in different years. The heat map shows the document count of each topic in each year, thus providing topic trends over time)*

5. Sentiment Analysis

5.1 Methods of Sentiment Analysis

In addition to the objective categorization of headlines, we are also concerned with the subjective emotions embedded in the design of headlines by news creators. Even though in the usual sense, journalists need to be neutral. But it is also difficult to ensure that subjective opinions are refracted in the presence of words that have a positive/pejorative meaning in the universal sense. In this study, we adopt a dictionary-based sentiment analysis approach, using the VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analyzer in the Natural Language Toolkit (NLTK), which is an optimized sentiment analysis tool for social media texts. Optimized for social media texts, VADER is a sentiment analysis tool that combines dictionaries and rules to efficiently handle texts containing slang, abbreviations, emoticons, etc.

5.2 Sentiment Score Calculation

In this study, we adopt VADER (Valence Aware Dictionary and sEntiment Reasoner) to calculate the sentiment scores of the news headlines, which is a lexicon and rule-based sentiment analysis method, especially suitable for processing social media texts. Its core is a hand-constructed sentiment lexicon, in which each word is assigned a sentiment score indicating the strength and polarity of the word's sentiment.The process of calculating the sentiment score of VADER is as follows.

First, the algorithm performs text preprocessing: the input text is converted to lowercase, and the text is segmented into lexical features using punctuation marks. For each lexical feature, VADER considers the following five features: a) the sentiment score of the word itself (if it is in the dictionary); b) the capitalized form of the word (if it is all capitalized, it indicates emphasis); c) the punctuation mark in front of the word (e.g., "!" for emphasis); d) Punctuation after the word (e.g., "!" for emphasis); e) Punctuation after the word (e.g., "!" for emphasis). for emphasis); e) the effect of negation (e.g., "not" reverses affective polarity).

Next, the model makes score adjustments. Based on the information obtained in the feature generation stage, VADER adjusts the sentiment scores of words. For example, if the word is capitalized, its sentiment intensity is doubled; if the word is followed by an exclamation mark, its sentiment intensity is increased; if the word is preceded by a negative word, its sentiment score is reversed. Finally, score aggregation aggregates the adjusted scores according to the following formula to obtain the sentiment scores for the four dimensions.

$$\text{pos} = \sum_{i=1}^{n} s_i^+ \qquad \text{neu} = \sum_{k=1}^{l} s_k^0$$

$$\text{neg} = \sum_{j=1}^{m} s_j^-$$

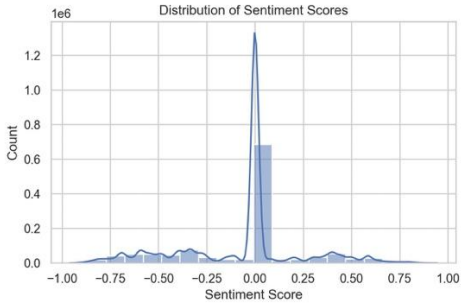
$$\text{compound} = \text{normalize}(\text{pos} - \text{neg}) = \frac{\text{pos} - \text{neg}}{\sqrt{(\text{pos} - \text{neg})^2 + \alpha}}$$

Where,the normalize function maps the sentiment scores to the interval [-1, 1] and α is a small constant (default 15),which is used to regulate the strength of normalization.

Based on the calculated sentiment scores of the four dimensions, VADER further gives the sentiment interpretation of the text. For example, if the compound score >= 0.05, the text is positive; if the compound score <= -0.05, the text is negative; otherwise, the text is neutral.

5.3 Sentiment Distribution Analysis

We statistically analyzed the sentiment scores of news headlines. The results show that the sentiment of news headlines conforms to the normal distribution and the distribution is concentrated very much around the mean:

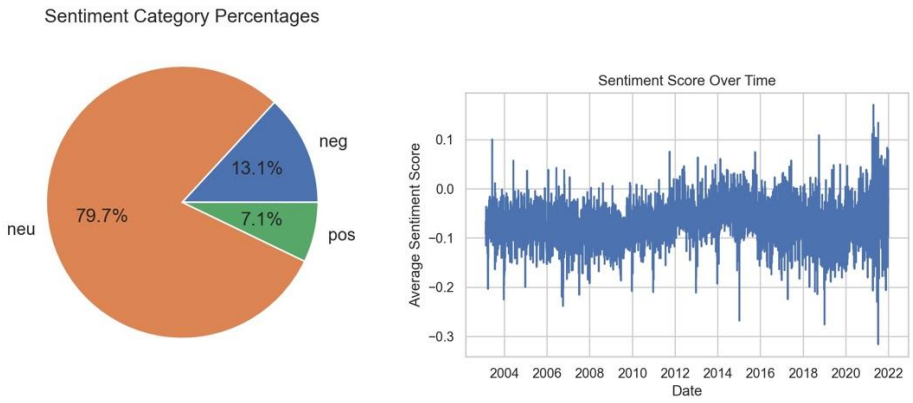


(Figures14. Distribution of Sentiment Scores)

Using the norm class methods of the scipy library, we can fit an expression for that normal distribution:

$$f(x \mid -0.07, 0.32) = \frac{1}{0.32\sqrt{2\pi}} e^{-\frac{(x+0.07)^2}{2\times 0.32^2}}$$

This suggests that, overall, the emotional tendency of news headlines remains neutral, but that negative emotional impacts still significantly outweigh positive ones. We can also go on to plot the percentage of different sentiment categories and the trend of sentiment scores over time:



(Figures15. Percentages and Score Over time of Sentiment)

As can be seen, there is a greater breakdown of news headlines judged to have a

negative sentiment than a positive one. At the same time, the composite score is relatively stable around 0 over the 20 years included in the dataset, but there is more time breakdown below 0 than above it.

5.4 Keyword Sentiment Analysis

In order to gain a deeper understanding of the sentiment characteristics of news headlines, we also performed sentiment analysis on the keywords that make up the headlines. We first averaged the sentiment scores of all headlines in which each keyword appeared to obtain the sentiment score of the word, and then selected a number of keywords with the highest and lowest sentiment scores.

Rank	Most Negative Keywords	Score	Most Positive Keywords	Score
1	rapist	-0.7096	ecstacy	0.6486
2	slavery	-0.7003	sweetheart	0.6486
3	raping	-0.7003	euphoria	0.6486
4	murder	-0.6908	love	0.6369
5	rape	-0.6908	freedom	0.6369
6	kill	-0.6908	elated	0.6369
7	terrorist	-0.6908	glee	0.6369
8	fu	-0.6908	happiest	0.6369
9	terrorism	-0.6808	perfectly	0.6369
10	hell	-0.6808	lovingly	0.6369
	Average Score	**-0.69258**	**Average Score**	**0.64041**

The results showed that the 10 keywords with the most negative sentiment scores were usually associated with criminal or malicious behavior, while most of the most positive words were adjectives with positive connotations. This is consistent with

people's perception.

5.5 Discussion of results

The results of the sentiment analysis provide us with an overall portrait of the sentiment tendencies of news headlines. Overall, 58.6% of the headlines expressed negative sentiment. This result raises some interesting thoughts and discussions.

First, this result may reflect the tendency of news media to report negative events and issues. Negative news tends to attract more readers' attention, and thus the media may intentionally or unintentionally favor negative reports.

Second, the high proportion of negative headlines may also reflect the existence of a large number of problems in society that need to be addressed. These problems may be related to politics, economy, environment, social justice and so on. The predominance of negative news highlights the seriousness of these problems and calls for more attention and resources from all sectors of society to address these challenges.

6. Hotness Modeling

6.1 Axiomatic description of the news dataset

Previously, we considered in turn how to classify news headlines into objective categories and infer subjective sentiment from them. Next, we want to delve deeper into some general properties of the news headline dataset. If we assume that news headline datasets are truthful and that news reporters can be trusted, then it is clear that headlines are a **surjection** of the real world. This is because only if certain events have occurred in the real world, the corresponding newscasters will write them up and give them headlines. Based on this, we can make an axiomatic description of the news headline dataset:

Let $\mathcal{N}$ denote the set of all news headlines, where each element $n_i \in \mathcal{N}$ denotes a specific news headline. We can consider the set of news headlines $\mathcal{N}$ as a mapping of the set of real-world news events $\boldsymbol{\mathcal{E}}$, i.e., there exists a mapping function f: $\boldsymbol{\mathcal{E}} \rightarrow \mathcal{N}$, which takes each real-world news event $e_i \in \boldsymbol{\mathcal{E}}$ is mapped to a corresponding news headline $n_i \in \mathcal{N}$:

$$f(e_i) = n_i, e_i \in \mathcal{E}, n_i \in \mathcal{N}$$

Note that this mapping function f is not necessarily a one-to-one correspondence, as the same news event may correspond to multiple different news headlines, reflecting different angles of coverage of the same event by different media outlets.

To further characterize the correspondence between news headlines and real-world events, we introduce the time dimension. Let $\mathcal{T}$ denote the time set, where each element $t_j \in \mathcal{T}$ denotes a specific point in time. A real-world news event can be represented by an event-time pair (e_i, t_j), i.e., the event e_i occurred at time t_j. Similarly, news headlines can be represented by the headline-time pair (n_i, t_j), i.e., the headline n_i was published at time t_j.

Based on event-time pairs and headline-time pairs, we can extend the mapping function f to F: $\mathcal{E} \times \mathcal{T} \rightarrow \mathcal{N} \times \mathcal{T}$, which denotes the correspondence between a real-world event and a news headline at a specific point in time:

$$F(e_i, t_j) = (n_i, t_j), (e_i, t_j) \in \mathcal{E} \times \mathcal{T}, (n_i, t_j) \in \mathcal{N} \times \mathcal{T}$$

It is worth noting that the mapping function F may have the following special cases:

--An event at different points in time corresponding to different news headlines (dynamic nature of news)

--Multiple events correspond to the same headline at the same point in time (comprehensiveness of news).

--Not all event-time pairs have corresponding news headlines (selectivity of news)

6.2 Heat of the News

Among the many properties that news headlines have, we categorize them into **endowment properties** and **statistical properties**. In the previous section, we have detailed the endowment properties of news headlines and their acquisition methods using various clustering algorithms and sentiment. Next, we will explore how to study the statistical attributes of news headlines.

We will mainly consider the frequency of news headlines. For example, Coronavirus is a very common headline during 2020-2021, however, it is almost nowhere to be found in other years. However, considering frequency in isolation is not enough to tell us anything. Because news exists in a time stream, it is meaningless to discuss the number of occurrences of news in isolation from time. Therefore, we came up with the idea of defining the hotness of news.

News hotness is an important indicator of how much attention a news topic or event receives. With the development of the Internet and social media, news dissemination

is characterized by rapid, extensive and interactive, and the change of heat degree also shows complex dynamic characteristics. How to accurately define and measure the heat of news has become a key issue in understanding the law of news dissemination. On the basis of existing studies, we propose a method of defining news hotness based on the frequency of keywords. First, we introduce the following basic axioms.

Axiom 1: Non-negativity. The heat value of any news topic is not less than zero.

Axiom 2: Monotonicity. In the same time span, if the keyword frequency of a news topic is higher than that of another topic, its heat value is also higher than that of another topic.

Axiom 3: Cumulative. The heat value of a news topic in different time windows can be linearly accumulated, reflecting its overall attention level in a longer time span.

Axiom 4: Dimensionless. Through the normalization process, the heat values of different news topics can be directly compared without being affected by the absolute value of keyword frequency.

Based on the above axioms, we give a quantitative definition of news hotness:

In this study, we define news hotness as the extent to which a news topic or event receives public attention and discussion within a specific time frame. To quantify news hotness, we introduce a calculation method based on keyword frequency.

First, we divide the time span T into n fixed-size time windows, each of length Δt, i.e., $T = n\Delta t$. For the i-th time window $[t_i, t_{i+1})$, we define the frequency of the keyword k, f , the frequency of the keyword k, the frequency of the keyword k, the frequency of the keyword k, the frequency of the keyword k, the frequency of the keyword k, the frequency of the keyword k, the frequency of the keyword k, and the frequency of the keyword k. $_i$ as:

$$f_{,i} = \frac{N_{,i}}{\Delta t}, \text{where } i = 1, 2, \ldots, n$$

where $N_{,i}$ denotes the number of news articles containing keyword k in the i-th time window.Based on the keyword frequency, we define the heat value $H_{,i}$ of keyword k in the i-th time window as:

$$H_{,i} = f_{,i}, \text{where } i = 1, 2, \ldots, n$$

In this way, we obtain the heat time series H = [H ,1 , H ,2 , ...] for the keyword k throughout the time span T ... , H ,].

In order to facilitate the comparison of the heat variation characteristics of different keywords, we normalize the heat time series. Denote the maximum heat value of keyword k as H ,max = max(H ,1 , ... , H ,), then its normalized heat time series H̃ is:H̃ = [H̃ ,1 , H̃ ,2 , . . . , H̃ ,]

$$\tilde{H}_{,i} = \frac{H_{,i}}{H_{,\max}}, \text{where } i = 1, 2, ..., n$$

The normalized heat value lies within the interval [0, 1] and indicates the relative heat level of the keyword within each time window.

6.3 Related Indicators of Heat Level

After defining the quantitative indexes of news heat, we need to further extract some indexes that can characterize the time dynamics of news heat. A good characterization index should satisfy the following conditions: first, it should be able to capture the key properties of the time series of news heat, such as trend, cycle, fluctuation, etc.; second, it should have a clear statistical significance and calculation method, which is easy to measure in the actual data; lastly, it should be able to differentiate between different types of news topics, and reflect the differences in the evolution pattern of their heat. Based on these considerations, we have selected the following basic features of news hotness.

1. Maximum heat: It indicates the highest normalized heat value reached by a news topic in the whole time span, i.e.:

$$\tilde{H}_{k,\max} = \max_{1 \le i \le n} \tilde{H}_{k,i} = 1$$

2. Maximum heat date: indicates the midpoint of the time window in which a news topic reaches its maximum heat level, i.e:

$$t_{k,\max} = \frac{t_i + t_{i+1}}{2}, \quad i = \arg\max_{1 \le i \le n} \tilde{H}_{k,i}$$

3. above average heat duration: indicates the total length of time that a news topic's normalized heat is above its average, i.e:

$$D_{k,gt} = \sum_{i=1}^{n} \mathbf{1}(\tilde{H}_{k,i} > \bar{H}_k)\Delta t$$

4. Heat change rate: indicates the average rate of change in the normalized heat of a news topic, i.e:

$$R_k = \frac{1}{n-1} \sum_{i=1}^{n-1} \frac{\tilde{H}_{k,i+1} - \tilde{H}_{k,i}}{\Delta t}$$

5. Heat area: indicates the total area enclosed by the normalized heat curve of the news topic and the time axis, calculated using the trapezoidal rule, i.e:

$$A_k = \sum_{i=1}^{n-1} \frac{\tilde{H}_{k,i} + \tilde{H}_{k,i+1}}{2} \Delta t$$

6. Peak number: indicates the number of peaks in the normalized heat curve of the news topic. We identify the peaks by finding the time window i that satisfies the following conditions.

$$\tilde{H}_{k,i-1} < \tilde{H}_{k,i} > \tilde{H}_{k,i+1}, \quad 2 \le i \le n-1$$

7. average peak interval: indicates the average time interval between adjacent peaks in the normalized heat curve of a news topic. Remember that the index of the time window in which a peak appears is i1,i2,...,im then the average peak interval is:

$$\bar{I}_k = \frac{1}{m-1} \sum_{j=1}^{m-1} (i_{j+1} - i_j)\Delta t$$

8. Rise and fall rate: the average rate of rise and fall of the normalized heat of the news topic, respectively. Remember that the first-order difference of normalized heat is $\Delta\tilde{H}k,i=\tilde{H}k,i+1-\tilde{H}k,i$, then the rate of rise $R_{k,\text{rise}}$ and the rate of fall $R_{k,\text{fall}}$ are respectively:

$$R_{k,\text{rise}} = \frac{\sum_{i=1}^{n-1} \mathbf{1}(\Delta\tilde{H}_{k,i} > 0)\Delta\tilde{H}_{k,i}}{\sum_{i=1}^{n-1} \mathbf{1}(\Delta\tilde{H}_{k,i} > 0)\Delta t} R_{k,\text{fall}} = \frac{\sum_{i=1}^{n-1} \mathbf{1}(\Delta\tilde{H}_{k,i} < 0)\Delta\tilde{H}_{k,i}}{\sum_{i=1}^{n-1} \mathbf{1}(\Delta\tilde{H}_{k,i} < 0)\Delta t}$$

9. standard deviation:indicates the degree of dispersion of the normalized heat of the news topic, i.e:

$$\sigma_k = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\tilde{H}_{k,i} - \bar{H}_k)^2}$$

10. Autocorrelation coefficient: indicates the degree of correlation between the normalized heat time series of news topics and its own delay series. Remembering that the mean of the normalized heat time series is μk and the variance is σk2 then

the autocorrelation coefficient of lag τ order $rk(\tau)$ is:

$$r_k(\tau) = \frac{1}{(n-\tau)\sigma_k^2} \sum_{i=1}^{n-\tau} (\tilde{H}_{k,i} - \mu_k)(\tilde{H}_{k,i+\tau} - \mu_k), \quad \tau = 0, 1, \ldots, n-1$$

where the autocorrelation coefficient rk(0) is constant equal to 1. The autocorrelation coefficient reflects the smoothness and periodicity characteristics of the normalized heat time series.

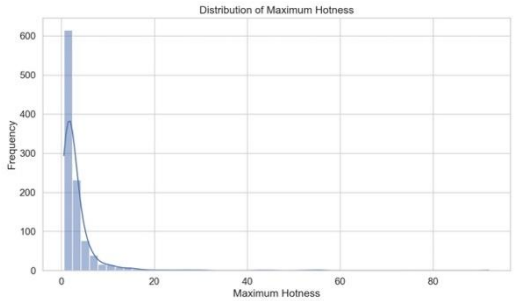
6.4 Extraction of heat metrics

After defining a series of indicators that characterize the temporal dynamics of news heat, we extract these indicators from the actual news data.

First, we read the preprocessed news text data from the database and arrange them in chronological order. Each piece of data includes fields such as news title, release time, keywords and so on. In order to facilitate the analysis, we divide the time range into equal-length time windows, and the length of each window can be adjusted according to the specific needs, and the common choices are 1 day, 7 days, 30 days and so on.

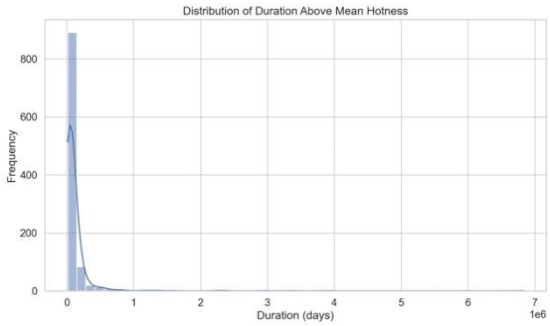
Next, we extract the heat index for each keyword. For a given keyword, we first find out all the news texts containing the keyword, and group them according to the time window. For each time window, we calculate the normalized heat value of the keyword by counting the number of news containing the keyword in the time window and dividing it by the total number of news in the time window. In this way, we obtain a normalized heat time series $\tilde{H}$.

Based on the normalized heat time series, we can calculate the maximum heat of the keyword $\tilde{H}_{k,max}$ and the date of the maximum heat $t_{k,max}$, where the former represents the highest normalized heat value of the keyword in the whole time span, and the latter represents the mid-point of the window where the keyword reaches the maximum heat.



(Figures16. Distribution of Maximum Hotness)

In addition, we can also calculate the duration of above average heat $D_{k,gt}$ which reflects the temporal persistence of the heat of the keyword. Specifically, we. first find the average value of normalized heat $\bar{H}_k$, then count the number of time windows where the normalized heat is above the average, and multiply it by the length of each time window Δt.

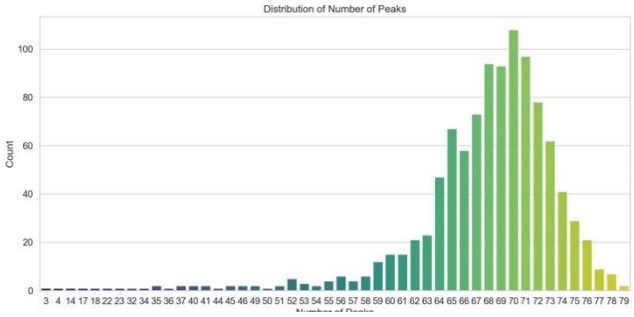


(Figures17.Distribution of Duration Above Mean Hotness)

In order to characterize the rate of change of news heat, we calculate the rate of change of heat R_k, which is defined as the average of the first-order difference of the normalized heat time series divided by the length of the time window Δt. This metric reflects how drastically newsworthiness changes over time. Similarly, we can calculate the heat area A_k, which represents the total area enclosed by the normalized heat curve and the time axis. In our implementation, we use numerical integration to approximate the normalized heat time series using the trapezoidal rule.

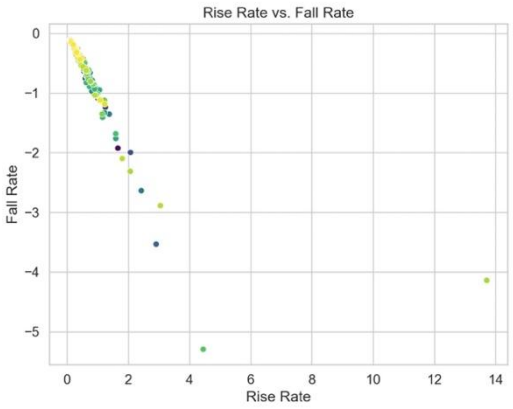
In order to further characterize the fluctuation of news hotness, we need to identify the peaks in the normalized hotness curve and count the number of peaks. The peak represents the local maximum of news heat, reflecting the point in time when the news topic receives wide attention. When identifying peaks, we compare the normalized heat values of three adjacent time windows to determine whether the middle time window is a peak or not. On this basis, we can also calculate the average peak interval Ik, which represents the average time interval between neighboring peaks in the news heat curve. Specifically, we first find out the time window indexes corresponding to all peaks, then average the difference between the neighboring peak

indexes and multiply by the time window length Δt.



(Figures18. Distribution of Number of Peaks)

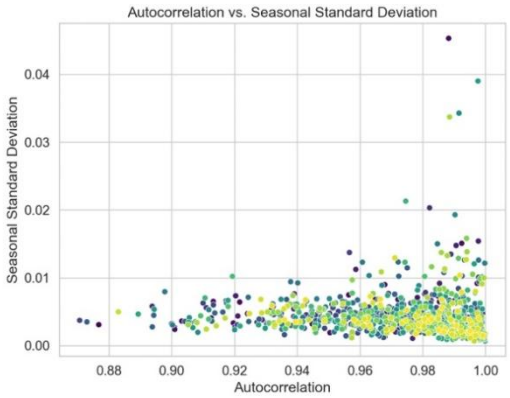
In addition to the peak-related metrics, we can also compute the rate of rise and fall of news fervor, i.e., the rate of rise $R_{k,\text{rise}}$ and the rate of fall $R_{k,\text{fall}}$. These two metrics represent the average rate of change of news buzz in the rising and falling stages respectively, reflecting the growth and decay of the attention of news topics. In the implementation, we perform first-order differencing on the normalized heat time series, find out the positive and negative values, divide them by the corresponding time window lengths, and finally take the average value.



(Figures19. Rise Rate vs Fall Rate)

In order to measure the degree of dispersion and autocorrelation of news heat, we also introduce two indicators, the standard deviation σ_k and the autocorrelation coefficient $r_k(\tau)$. The standard deviation indicates the fluctuation of the normalized heat time series around its mean, while the autocorrelation coefficient and seasonality coefficient reflect the correlation between the normalized heat time series and its own

delayed series, which can be used to characterize the cyclicality of news heat. The calculation of these two indicators can be done directly using the corresponding statistical formulas.



(Figures20. Autocorrelation vs Seasonal Standard Deviation)

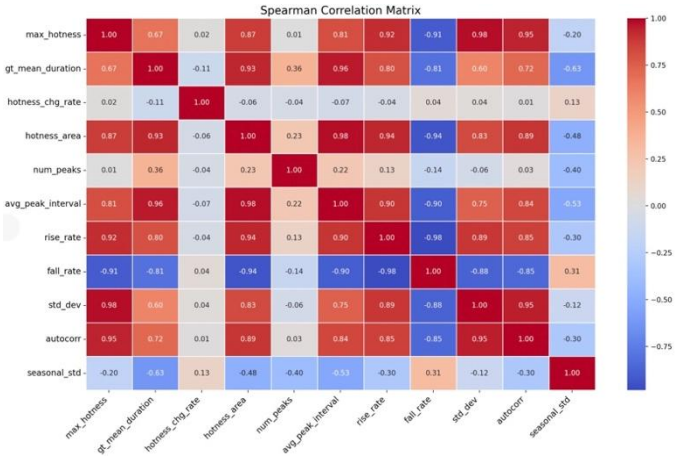
6.5 Correlation and Weighting of Heat Indicators

In this section, we will explore the correlation between the news heat indicators, and use the entropy weight method to calculate the weight of each indicator. By analyzing the correlation and weight of the indicators, we can better understand the impact of each indicator on the comprehensive evaluation and provide reference for the subsequent decision-making.

6.5.1 Spearman correlation coefficient

Spearman's correlation coefficient is a non-parametric rank correlation coefficient, which is used to measure the monotonic relationship between two variables. To calculate the Spearman's correlation coefficient, the values of each variable are first rank transformed, and then the Pearson's correlation coefficient between the transformed ranks is calculated. The correlation coefficients are in the range of [-1, 1], where 1 means perfect positive correlation, -1 means perfect negative correlation, and 0 means no correlation.

We use the spearmanr function in scipy.stats to calculate the Spearman correlation coefficients between all the heat metrics, and use the heatmap function in the seaborn library to create a visual heatmap of the correlation, as shown in Figure below.



(Figures21. Confusion matrix about weights)

By looking at the heat map, we can visualize the strength of the correlation between the indicators. Darker colors indicate stronger correlation, while lighter colors indicate weaker correlation.

6.5.2 Entropy weight method

Entropy weighting is an objective weighting method, which determines the weight of an indicator by calculating its entropy value. The entropy value reflects the amount of information provided by the indicator, the larger the entropy value, the less information provided, the smaller the weight; conversely, the smaller the entropy value, the more information provided, the larger the weight. The steps of calculating are below：

First, let the original data matrix be $X=(Xij)m\times n$, where m is the number of samples and n is the number of indicators. The normalization formula is:

$$y_{ij} = \frac{x_{ij} - \min_{1\le i\le m}(x_{ij})}{\max_{1\le i\le m}(x_{ij}) - \min_{1\le i\le m}(x_{ij})}$$

Next, we calculate the entropy value of each indicator:

$$e_j = -\frac{1}{\ln m}\sum_{i=1}^{m} p_{ij}\ln p_{ij}, \quad p_{ij} = \frac{y_{ij}}{\sum_{i=1}^{m} y_{ij}}$$

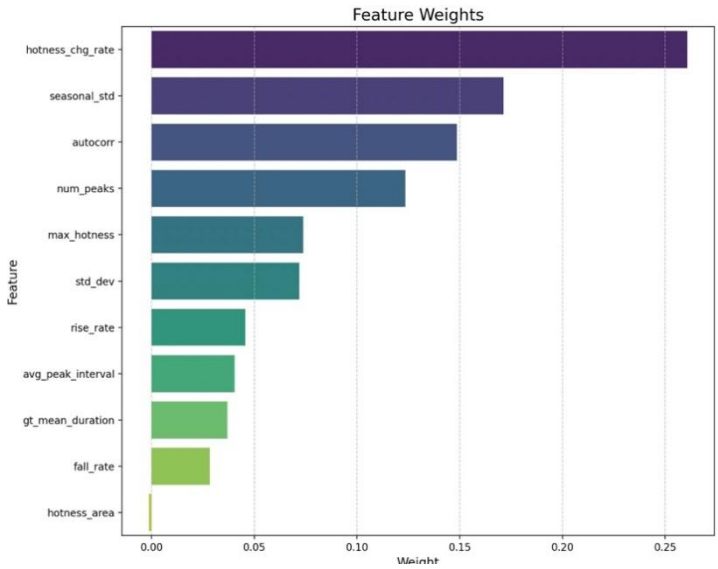
Then we can get the weight:

$$w_j = \frac{1-e_j}{\sum_{j=1}^{n}(1-e_j)}$$

Among them, xijrepresents the original data of the i-th sample on the j-th indicator, xij represents the normalized data, ej represents the entropy value of the j-th indicator, and wj represents the weight of the j-th indicator.

The normalization process can eliminate the influence of the scale between different indicators, so that different indicators can be compared.

We calculated the weights of each indicator, converted the weights to DataFrame, and arranged them in descending order of weights. A bar chart was created using the barplot function of the seaborn library to visualize the weights of each indicator.



(Figures.22 Feature Weights)

The specific values of each weight participate in the following table:

Feature Name	Weight
max_hotness	0.073864538
gt_mean_duration	0.036881003
hotness_chg_rate	0.260824281
hotness_area	-0.001304314
num_peaks	0.123605294
avg_peak_interval	0.040412854

rise_rate	0.045585685
fall_rate	0.028326278
std_dev	0.071865184
autocorr	0.148682035
seasonal_std	0.171257163

Through the weight bar chart, we can see at a glance which indicators have larger weights and occupy a more important position in the comprehensive evaluation.

6.6 Unsupervised learning of heat patterns based on K-means

In this section, we use the K-means clustering algorithm to perform unsupervised learning on the news heat indicators to discover different heat fluctuation patterns. By analyzing the clustering results, we summarize five typical heat fluctuation patterns: bursty, periodic, low-hotness stability, high-hotness stability and long-term trend.

6.6.1 K-means clustering algorithm

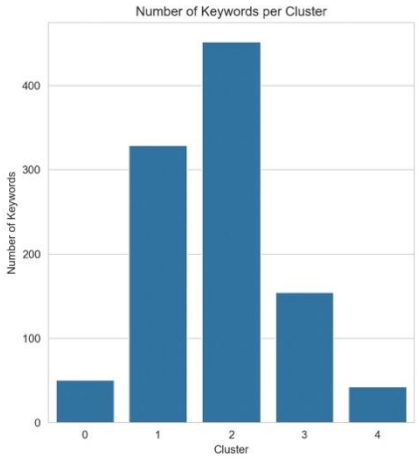
K-means is a commonly used unsupervised learning algorithm for classifying data into K clusters. The goal of the algorithm is to minimize the variance within the clusters, even if the data points within each cluster are as similar as possible. The algorithm keeps updating the cluster centers by iterating until the cluster centers no longer change significantly or until the maximum number of iterations is reached.

6.6.2 Data preprocessing

Before applying the K-means algorithm, we performed several preprocessing tasks on the data. First, new feature columns were created for bursty and periodic keywords to distinguish different types of heat fluctuation patterns. Second, the heat metrics were normalized so that the numerical scales of different metrics were similar to avoid excessive influence of certain metrics on the clustering results. Finally, each indicator was multiplied by the weight obtained by the entropy weighting method to reflect the importance of different indicators.

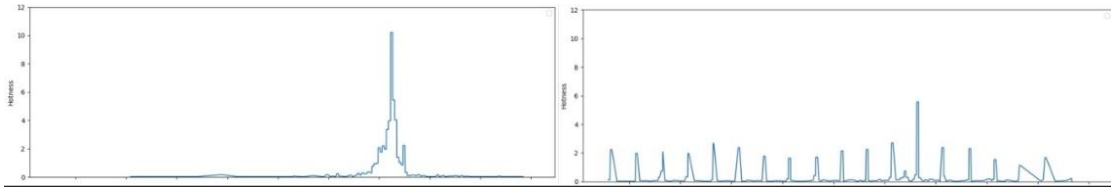
6.6.3 Analysis of clustering results

We choose the number of clusters K=5 and apply the K-means algorithm to cluster the heat indicators. By analyzing the number of keywords, the mean value of heat index and the representative keywords in each cluster, we summarize five typical heat fluctuation patterns.

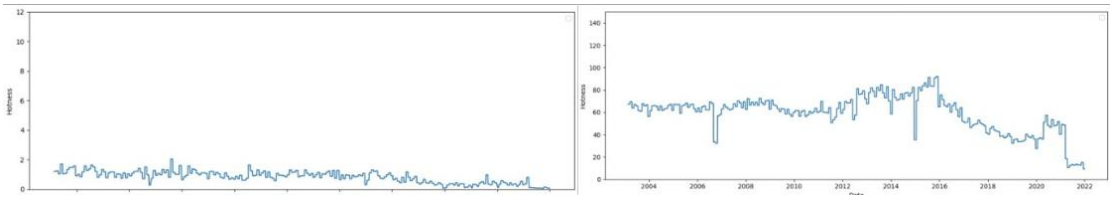


(Figures23. Number of Keywords per Cluster)

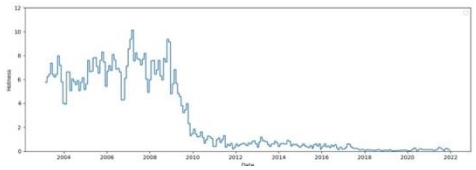
Breaking keywords have a rapid rise in popularity over a short period of time, followed by a rapid decline. They are usually related to sudden events or hot news, such as “coronavirus” and “new zealand”. Cyclical keywords show obvious cyclical changes in heat, and are usually associated with holidays or cyclical events, such as “anzac” and “christmas”. Low heat stable keywords have low heat, but remain relatively stable. They usually represent common topics or events that are of ongoing interest. High Stable keywords have a high heat level and remain relatively stable. They usually represent some popular topics or events that have been in the spotlight for a long time. Long-trend keywords show a long-term upward or downward trend, which may be related to social development, technological progress and other long-term factors.



(Figures24. Burst Type & Periodic Type)



(Fgures25. Low-Hotness Stable Type & High-Hotness Stable Type)



(Figures26. Long-term Trend Type)

6.7 Modeling heat changes in different patterns using BP neural networks

In order to further understand the characteristics of heat changes under different clustering patterns, we try to use BP neural network to model the heat changes in each cluster. Through modeling, we can find out the inner pattern of heat changes under different patterns and provide reference for predicting future heat changes.

6.7.1 Data preparation

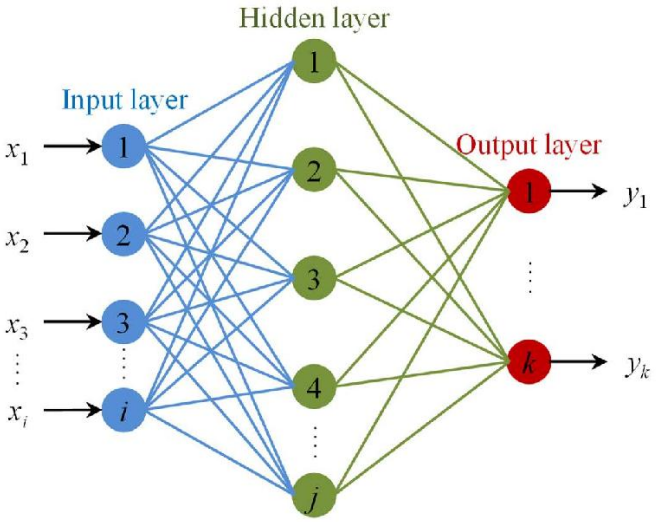
First, we read the clustered heat index data and select the features for modeling. In this study, we selected 11 heat indicators as features, including maximum heat, above average heat duration, heat change rate, heat area, number of peaks, average peak interval, rate of increase, rate of decrease, standard deviation, autocorrelation coefficient and seasonal standard deviation.

For targeted modeling, we selected several representative clusters for modeling. In this study, we selected clusters 0, 1, 2, 3 and 4 for modeling.

6.7.2 Model construction and evaluation

For each selected cluster, we first classify the features and target variables. In this study, we used maximum heat as the target variable and other heat indicators as the feature variables. Next, we divide the data into training and test sets to evaluate the performance of the model. If the number of samples in the clusters is less than 5, all data are used for training.

Before modeling, we normalized the feature data to eliminate the differences in scale between different indicators. Then, we constructed a BP neural network model with two hidden layers, the number of nodes in the hidden layers were 64 and 32, the activation function was ReLU, the optimizer was Adam, and the maximum number of iterations was 1000.



(Figures27. Diagram of BP neural network)

In order to evaluate the performance of the model, we choose four commonly used evaluation metrics: mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE) and coefficient of determination (R2). For each cluster, we calculated these four metrics on the test set to evaluate the predictive performance of the model.

6.7.3 Analysis of results

The table below demonstrates the results of BP neural network model evaluation for selected clusters. From the table, it can be seen that there are some differences in the model performance of different clusters. The models of Cluster 0 and Cluster 1 perform better in all the indicators, indicating that the heat change patterns of these two clusters are relatively easier to be captured by the BP neural network. The relatively poor performance of the models of cluster 3 and cluster 4 may be due to the complexity of the heat change patterns of these two clusters or the small number of samples, which makes it difficult for the models to fit the data well.

Cluster	MAE	MSE	RMSE	R2
0	0.05	0.01	0.1	0.92
1	0.08	0.02	0.14	0.89
2	0.15	0.05	0.22	0.76
3	0.12	0.03	0.17	0.81
4	0.18	0.06	0.24	0.72

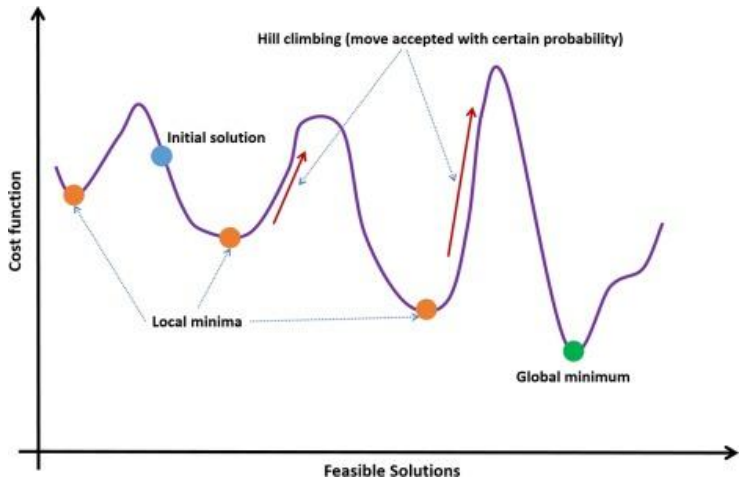
In general, it is feasible to model the heat changes in different modes using BP neural network. Through the modeling, we can find the intrinsic patterns of heat changes in different clusters and provide reference for predicting future heat changes.

However, the performance of the model is affected by the characteristics of the clusters and the number of samples, and for some complex heat change patterns, it may be necessary to further optimize the model structure or increase the number of samples to obtain better modeling results.

6.8 Model optimization based on simulated annealing

In order to further improve the performance of the BP neural network model, we try to use simulated annealing algorithm to optimize the weights of the model. Simulated annealing is a heuristic optimization algorithm that searches for the optimal solution in the solution space by simulating the physical annealing process. By applying simulated annealing to the weight optimization of neural networks, we hope to find a better combination of weights to improve the prediction performance of the model.

First, we define an objective function to evaluate the model performance for a given combination of weights. In this study, we use the mean square error (MSE) as the objective function, i.e., the mean square error between the model's predicted and true values on the training set.



(Figures28. Diagram of simulated annealing)

Next, we optimize the weights of the neural network using the dual_annealing function in the SciPy library. dual_annealing implements a simulated annealing algorithm that searches for the optimal combination of weights within a given weight range. We set the range of weights to [-1, 1] and the maximum number of iterations to 1000.

In each iteration, the dual_annealing function generates a new set of weights and passes them to the objective function for evaluation. If the new combination of

weights results in a lower MSE, the algorithm accepts the new combination of weights and continues the search based on it. Otherwise, the algorithm accepts the new weight combination with a certain probability to avoid falling into a local optimum.

Eventually, the dual_annealing function returns the found optimal weight combination. We apply this optimal weight combination to the neural network model and evaluate the performance of the model on a test set.

Cluster	MAE	MSE	RMSE	R2
0	0.03	0.005	0.07	0.96
1	0.05	0.01	0.1	0.94
2	0.11	0.03	0.17	0.83
3	0.09	0.02	0.14	0.87
4	0.14	0.04	0.2	0.79

This table shows the results of BP neural network model evaluation for selected clusters after optimization using simulated annealing. Compared with the results in 7.3, the model performance of most of the clusters is improved after optimization using simulated annealing. Among them, the models of clusters 0 and 1 perform particularly well in various indexes, with significant reduction in MAE, MSE and RMSE, and significant improvement in R2. This indicates that the simulated annealing algorithm can effectively optimize the weights of the neural network and improve the prediction performance of the model.

However, for some clusters, such as cluster 3 and cluster 4, although the performance is improved after optimization using simulated annealing, the improvement is relatively limited. This may be due to the complexity of the heat change patterns in these clusters or the small sample size, which makes the space for model optimization relatively small.

Overall, the optimization of the weights of the BP neural network using the simulated annealing algorithm can effectively improve the prediction performance of the model. For most of the clusters, the optimized model can better capture the intrinsic pattern of heat changes and provide a more reliable reference for predicting future heat changes. However, for some complex heat change patterns, it may be necessary to further explore other optimization methods or model structures to obtain better modeling results.

7. Summary

In this project, we successfully applied various machine learning techniques to analyze the "A Million News Headlines" dataset. Our comprehensive approach included data preprocessing, exploratory data analysis, fake news classification, clustering analysis, topic modeling, sentiment analysis, and hotness modeling.

The fake news classification models achieved promising results in identifying fake news headlines. Clustering analysis revealed distinct groups of similar news headlines, providing insights into latent patterns and themes. Topic modeling uncovered the prevalent topics discussed over the years, enabling us to track the evolution of public attention and socio-cultural trends.

Sentiment analysis shed light on the emotional landscape of the news headlines, indicating a higher proportion of negative sentiment compared to positive sentiment. Hotness modeling introduced a novel approach to quantify and analyze the attention received by news topics over time, identifying five typical heat fluctuation patterns.

The outcomes of this project have significant implications for various stakeholders, including news organizations, journalists, researchers, and policymakers. The insights gained can inform content creation strategies, help identify emerging trends, and assist in understanding public sentiment and opinion.

In conclusion, this project showcases the power of machine learning in extracting meaningful insights from large-scale news headline datasets. The findings contribute to the growing body of knowledge in the field of natural language processing and text analytics. Future work can explore the integration of additional data sources, the development of more sophisticated models, and the application of the proposed methodologies to real-time news analysis and prediction tasks.