**Text Analytics Course Work**

**Abstract**

Task 1 uses unsupervised learning methods, SVM, random forests, oversampling, neural network models, LDA models and other data science practices to explore the natural language processing practice of emotion extraction text. Task 2 used methods such as the BERT model to explore the practical process of setting up NER and gained a lot.

**Task 1**

In **Task 1.1** I used two non-neural network models, the first one is TF-IDF (Term Frequency-Inverse Document Frequency) combined with SVM.TF-IDF is a method used for textual feature extraction, which measures the importance of words in a document and in the corpus as a whole, and the SVM model is a supervised learning model, which in this case uses hyperplanes to achieve discriminative classification. The second is Word2Vec combined with Random Forest, a model for generating word vectors. Although Word2Vec is based on neural networks, the use of a pre-trained Word2Vec model here does not involve re-training the neural network, but rather utilises existing word vectors.Random Forest is an integrated learning method based on decision trees, which improves the accuracy and robustness of the model by constructing multiple decision trees and either averaging or voting on their predictions.

Model 1 Before feature extraction, the text data needs to be cleaned by removing special characters, punctuation marks and stop words to reduce noise and irrelevant information. I removed all non-alphanumeric characters, all numbers, and also converted them to lowercase letters, which in turn led to text vectorisation using TF-IDF By calculating the Word Frequency (TF) and the Inverse Document Frequency (IDF), a weight is assigned to each word, reflecting the importance in the text. In order to limit the maximum number of features in TF-IDF vectorisation to reduce the dimensionality and avoid overfitting and excessive computational complexity use the 'max\_features' parameter for tuning, use the 'max\_iter' parameter logistic regression to ensure that the model can converge sufficiently. There is also a regularisation parameter for the logistic regression that is used to control the complexity of the model, where a smaller `C` value strengthens the regularisation and prevents overfitting, and a larger `C` value weakens the regularisation, so the appropriate value was chosen.

Model 2 is trained on the text data using the Word2Vec model to generate word vectors for each word. The pre-trained word vector model was used or further trained on a specific dataset. Document vectors are generated by averaging the word vectors in the document, and the document vectors are computed as input features for the classification model. The dimensionality of the word vectors, the size of the context window, and most critically, the number of trees in the random forest and the maximum depth of each tree in the random forest are set.

Also in classification tasks, the problem of class imbalance may cause the model to be biased towards predicting the majority class, thus affecting the performance of the model. To solve this problem, I used an oversampling method to balance the data, SMOTE (Synthetic Minority Over-sampling Technique). By generating new minority class samples, the class distribution is balanced so that the classification model can better learn the minority class features and improve the accuracy and recall of the model. Perform feature extraction and training of classification models on the over-sampled data. Adjustment of the scaling parameter `sampling\_strategy` which determines the generation of minority class samples, and `k\_neighbors` which is used to generate the number of nearest neighbors for new samples

The design and optimisation of these three models enables us to better understand the performance of different text feature extraction methods and classification models. In particular, by oversampling the optimisation model, we can effectively solve the category imbalance problem, improve the accuracy and recall of the model, and enhance the robustness of the classification model.

The training of the first model resulted in an accuracy of 0.66, and after improving the model with the addition of oversampling, the accuracy was still 0.66, which didn't change much. The experimental accuracy of the random forest was 0.6 and the second model had just 0.4. It was not very effective, but I learnt very interesting machine learning algorithms in practice.

**Task 1.2** Objectives required us to design algorithms using neural networks, I chose a Convolutional Neural Network (CNN) based architecture mainly because CNNs perform well in text classification tasks, especially in extracting locally relevant features in text. The Conv1D Layer is a one-dimensional convolutional layer used to extract local features from sequential data, the parameters were initially set using 64 convolutional kernels of size 5 each and an activation function of ReLU, with the aim of capturing local contextual relationships in the text. The Embedding Layer efficiently processes textual data by converting words into a more expressive vector form, converting word indices of textual inputs into dense vectors of fixed size. This layer maps each word to a high-dimensional space to capture the semantic relationships between words. GlobalMaxPooling1D is a global maximum pooling of the feature maps output from the convolutional layer, reducing the data dimensionality while retaining the most salient features. The Dropout layer is introduced to mitigate the risk of overfitting the model by randomly dropping some of the network connections to improve the model's generalisation ability The Dropout rate is set to 0.5, which is used to balance the complexity of the network with the training data. During the design process, the tuning of hyperparameters is very important and I have found various problems from them. MAX\_VOCAB\_SIZE (Vocabulary Size) and EMBEDDING\_DIM (Embedding Dimension) are used in order to compress the representation of words in the vocabulary while retaining enough information. There is also the Dense Layer which is a densely connected layer with a ReLU activation function to increase the nonlinearity of the network, as well as the Output Layer which is the final output layer using a SoftMax activation function for multi-classification problems.

After carefully going through several pre-processing experiments, I used a number of methods to optimise. For example, text cleaning was used to remove non-alphabetic characters such as punctuation marks and numbers from the text, which was used to reduce the noise. There is also lowercase conversion, which converts all text to lowercase and unifies the vocabulary to reduce word diversity. Sentences are broken down into words using a word-splitting tool. Deactivated words like "is", "and" etc. were removed as words usually do not carry useful information in text categorisation. I also used Tokenizer for vectorisation and pad\_sequences to ensure that all sequences are of the same length, converting the text to integer sequences so that they can be processed by the model.

A comprehensive analysis was performed as required by **Task 1.3**. The model was trained using the Adam optimiser with the EarlyStopping callback set to prevent overfitting. The initial learning rate was set to 0.001. EarlyStopping and ReduceLROnPlateau callback functions were used during training to monitor the validation loss to prevent overfitting and dynamically adjust the learning rate. The training accuracy rises as training proceeds, gradually increasing from an initial rate of about 0.4 to close to 0.9. This indicates that the model is gradually learning patterns and features from the training data. The validation accuracy rises rapidly in the initial phase, from about 0.4 to about 0.6, and then gradually levels off, eventually dropping slightly around 0.6. The rapid increase in validation accuracy indicates that the model effectively learnt the features in the data in the beginning phase. However, at a later stage, the growth of the validation accuracy slowed down and levelled off, and even had a slight decrease, which may indicate that the model is starting to overfit. So, I adjusted some hyper-parameter settings, such as learning rate, batch size, regularisation parameter, etc., and conducted experiments, but the results were still not good, even worse than not using the neural network learning model, which gave me a headache. This is because neural network models usually require a large amount of data to be fully trained and take advantage of them. If the amount of data is insufficient, the neural network model may not be able to learn enough features, resulting in a performance that is not as good as traditional non-neural network models. Although we used regularisation and early stopping strategies, overfitting still occurred when the model complexity was high, resulting in poor performance on the validation set. In the previous TF-IDF + SVM model, the TF-IDF extracted word frequency features, which are very effective for text classification tasks. The neural network model, on the other hand, uses embedding and convolutional layers, which may not use its full potential in feature extraction, especially when the amount of data is insufficient. The training curves show that the training and validation accuracies increase with epoch. A consistent increase in training accuracy indicates that the model is learning features from the training data.

**Task 1.4** Continuing to use the TweetEval sentiment dataset to simulate the identification of positive human emotions, I utilized the well-performing Potential Delicacy Assignment technique to help me achieve one of my aims. Firstly, I again utilized the technique of selecting tweets from the dataset that were labelled as 'optimistic' or 'happy'. The tagged values were viewed and the TF-IDF technique was used to identify the keywords or phrases in these tweets and the LDA (Linked Delicacy Allocation) topic modelling technique was used to identify and classify the topics based on the keywords. Latent Delicacy Allocation (LDA) is a generative probabilistic model that I can use well for topic modelling of text datasets, that's why the new approach was adopted instead of using machine learning or neural network models for the process because I wanted to try out a new practice that would make me more knowledgeable and familiar with Natural Language Processing. LDA assumes that a document consists of a number of topics, each of which is based on a set of words, which need to be identified and classified by the keywords. LDA assumes that a document consists of many topics, each of which is based on a set of words. By analyzing the common metrics of the words in the document, LDA needs to be able to automatically discover potential topic structures in the document set, and thus assign the document to different topics. These themes are then analyzed to help identify areas where people are often optimistic or happy, and the themes and their keywords are identified using the interactive interface provided by the pyLDAvis library. Here I have used pyLDAvis library based visualizations to display the themes, and the visualizations are very pleasing to the eye.

Detailed analysis of the themes generated by the LDA model revealed five primary themes with their associated vocabularies. Theme 1 focuses on the user experience related to watching live broadcasts and entertainment, featuring words like "watch", "live", and "broadcast" to capture users' positive responses to such events. Theme 2 revolves around daily life and emotional states, with terms like "day", "happy", and "love" highlighting discussions on daily activities and emotions. Theme 3 concerns user interaction and dialogue, using words such as "know", "get", and "good" to indicate inquiries and knowledge exchange. Themes 4 and 5, meanwhile, emphasize positive interactions and personal views, along with life events and celebrations, using words like "good", "love", "start", and "happy" to stress positive emotions and significant life moments.

Overall, these themes reflect a wide range of user interactions, expressions of emotion, and daily activities in the dataset. The frequent occurrence of the word "user" shows the user-centred nature of the data, while emotional words such as "happy", "love" and "horror" are used. "horror" demonstrate the range of emotions discussed. Words related to live streaming and entertainment appear in multiple themes, suggesting that these are important components of the data content. There are of course some drawbacks; the LDA model assumes that the words in the documents are independent, which is often inconsistent with real-world language use, as there are often semantic associations between words. And the performance of the model depends heavily on the choice of its hyperparameters (e.g., number of iterations and number of topics), and inappropriate parameter settings may lead to poor model performance. If it is in the actual language processing process can consider the use of Dynamic Topic Models (Dynamic Topic Models), this model takes into account the time-varying topic model, especially suitable for analysing text data that changes over time.

**Task 2**

Named Entity Recognition (NER) plays a crucial role in natural language processing and is mainly used to identify and classify entities with specific meanings from text, such as names of people, places and institutions. It is of great significance in information extraction, data organisation, knowledge graph construction and improving search efficiency. Especially in the medical field, NER technology can extract key medical information, such as diseases, symptoms and drugs, from medical records, clinical trial records and scientific literature, which is important for clinical decision support, accelerated medical research, disease monitoring and personalised medicine. Through NER, the efficiency and accuracy of medical information processing have been significantly improved, facilitating the advancement of medical research and healthcare services.

In the field of natural language processing, I was interested in the BERT model and wanted to try to explore it myself. BERT combines a pre-training and fine-tuning approach, where pre-training is performed on large-scale textual data and then fine-tuned to suit the specific task. Bi-directional encoders are able to consider the context of words simultaneously and can provide a deeper understanding of sentence structure and multiple meanings of words, thus significantly improving the accuracy of named entity recognition. However, there are some limitations to the use of BERT. Firstly, the model requires a large amount of computational resources, including GPUs and memory, especially when dealing with large datasets or training for a long period of time, which makes the model costly to train and deploy and may pose a challenge to researchers or developers with limited resources. Second, because BERT has a large number of parameters, it is prone to overfitting on small datasets. Although pre-training can partially mitigate this problem, the model may still tend to learn noisy rather than generalized features from the data, especially in tasks with small data volumes or imbalanced categories.

In this study, the PubMed "tner/bc5cdr" dataset containing sentences labelled with chemicals and diseases provided a comprehensive training environment. During pre-processing, I used "BertTokenizerFast", which is optimized for the BERT model, to segment words into lexical units necessary for BERT to process and understand natural language. This laxer effectively splits the text into sub-words, which is a key step in the structure of BERT. This approach is indeed simple enough, and I did not follow the script presented in the assignment optional because I wanted to explore a way to solve this problem on my own, which might have been more memorable. But the result was not ideal.

To solve the problem of the BERT classifier potentially breaking a single word into multiple sub-words, I implemented a tag alignment strategy. I specifically marked special tags such as [CLS], [SEP], and filler tags with unique labels (-100) so that the model would ignore these tags during training, which improves the accuracy of model recognition.

I adopted the Transformers-based BERT model, using a pre-trained "Bert-base-uncased" version of BERT, and added a token classification layer on top for the Named Entity Recognition (NER) task. I set the output size of this layer to match the five label categories needed for the task: "O" (non-entity), "B-Chemical" (beginning of chemical), "I-Chemical " (inside chemical), " B-Disease" (beginning of disease), and "I-Disease" (inside disease). This configuration allows BERT to process the input text and accurately classify entities, thus significantly improving the model's performance in extracting entities from biomedical text.

To continue optimising the performance of the BERT model, I set a series of training parameters: a training batch size of 16 to balance memory usage and computation time on different devices; an evaluation batch size of 64 to speed up the validation process; a 500-step warm-up step, which may be useful for adjusting the learning rate in the early stages of training to help the model avoid falling into a local optimum too early; and a weight decay of 0.01 to control the complexity of the model and prevent overfitting. , which is used to control the complexity of the model and prevent overfitting. In addition, I also set the learning rate adjustment strategy, which enables the model to automatically adjust the learning rate according to the validation results during the training process, so as to improve the training effect. These parameter configurations are based on pre-experimental results and best practices in the relevant literature, and by managing training at a fine-grained level, I effectively improve the accuracy of the model in recognising chemicals and disease entities in the biomedical literature.

图表, 折线图, 直方图

描述已自动生成

It has to be admitted that some gradient explosion can be seen in the graphs. Losses (shown in red) show fluctuations, although the fluctuations indicate that the learning process encounters variations in gradient decline due to the complexity of the dataset or small batch selection. The learning rate (shown in blue) clearly shows a triangular strategy - initially increasing to mid-training and then decreasing. This triangular learning rate scheduling or cyclic learning rate helps the model converge faster in the initial period and then fine-tune it in later periods. The peak in the learning rate roughly corresponds to the midpoint of training, which can be strategically planned to coincide with the period of maximum loss reduction. The sensitivity of the loss to changes in the learning rate indicates that the model's parameters are being significantly updated. and helps to get rid of local minima, which can be beneficial, but can also lead to training instability, as demonstrated by the loss surge.

In my next practice, I focused on exploring and comparing the efficacy of two textual similarity measures, particularly in terms of identifying the most and least similar entities to a particular complex disease entity. The method I chose was BERT and TF-IDF for comparative analyses.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Query Text** | **Method** | **Type** | **Similar/Dissimilar Entities** | **Similarity Score** |
| Tricuspid valve regurgitation and lithium carbonate toxicity in a newborn infant. | BERT | most similar | "Hepatic adenomas and focal nodular hyperplasia of the liver in young women on oral contraceptives" | 0.9865 |
| Tricuspid valve regurgitation and lithium carbonate toxicity in a newborn infant. | BERT | least similar | "Dexmedetomidine" | 0.1754 |
| Tricuspid valve regurgitation and lithium carbonate toxicity in a newborn infant. | TF-IDF | most similar | "A newborn with massive tricuspid regurgitation, atrial flutter, congestive heart failure, and a high serum lithium level is described." | 0.4215 |
| Tricuspid valve regurgitation and lithium carbonate toxicity in a newborn infant. | TF-IDF | least similar | "Water intoxication associated with oxytocin administration during saline - induced abortion." | 0 |

The entity with the most similar BERT results was "Hepatic adenoma and focal nodular hyperplasia of the liver in a young woman taking oral contraceptives: a case report." (similarity: 0.9865), while the least similar entity was "dexmedetomidine" (similarity: 0.1754).

The entity with the most similar TF-IDF results was "Describes a neonate with severe tricuspid regurgitation, atrial flutter, congestive heart failure, and high serum lithium levels." (Similarity: 0.4215), while the least similar entity was "Water intoxication associated with oxytocin administration during saline-induced abortion." (Similarity: 0.0).

The results show that BERT outperforms TF-IDF in recognising semantically rich, contextually relevant entities.BERT is able to understand deeper linguistic contexts and nuances in the text, thus providing higher similarity scores and more relevant matches. In contrast, TF-IDF, while useful for keyword-based similarity, is unable to capture the deeper semantic relationships required for complex medical texts, which is reflected in its lower similarity scores and fewer contextually relevant entity matches.