Multi-Label Classification on Research Articles Using Natural Language Processing Approach

Course : Text Mining
Lecturer : Lili Ayu Wulandhari, S.Si., M.Sc., Ph.D.

Written By:
Audrey Tabitha Ariani - 2440082812
Devin Augustin - 2440094352
Patrick Jonathan - 2440064791

Computer Science and Statistics
School of Computer Science
Bina Nusantara University

I. Introduction

A. Problem Statement

Researchers have access to large online archives of scientific articles. As a consequence, finding relevant articles has become more difficult. Tagging or topic modeling provides a way to give a token of identification to research articles which facilitates recommendation and search process.

The increasing volume of research articles published in diverse fields has led to a pressing need for effective methods for organizing and categorizing these articles. Multi-label classification, a type of machine learning problem that involves assigning multiple labels to a given input, has emerged as a promising approach to this problem. In this research paper, we explore the application of multi-label classification techniques for organizing research articles into multiple categories or topics.

The primary objective of this research paper is to evaluate the performance of various multi-label classification algorithms in predicting the topics of research articles, using a large and diverse dataset. We also investigate the impact of different feature representations, such as bag-of-words and word embeddings, on the classification accuracy.

B. Task

For this problem, we are going to do classification and prediction to solve this multi-label classification problem. Each sample data/text could be categorized as one or more from the following topics: (1) Computer Science, (2) Physics, (3) Mathematics, (4) Statistics, (5) Quantitative Biology, and (6) Quantitative Finance. We are going to predict if the article has one or more of these topics.

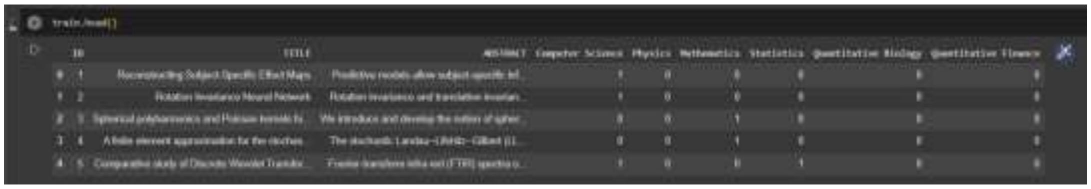
C. Simple Data Analysis

This data is taken from the "NLP on Research Articles" dataset. We use the data train.csv in the dataset to carry out the classification and prediction processes. In the data train.csv there are 20972 data where each data has :

- "Id"
- "Title" (title of the research article)
- "Abstract" (overview of the research article)
- "Computer Science" (topic label category)
- "Physics" (topic label category)
- "Mathematics" (topic label category)
- "Statistics" (topic label category)
- "Quantitative Biology" (topic label category)
- "Quantitative Finance" (topic label category)

Each topic label category will be filled with a binary number, where 0 means the observation is not included in that topic label, while 1 means the observation is included in that topic label. The process of classifying and predicting category labels will be carried out using Natural Language Processing on the variables "Title" and "Abstract".

Data overview :



```
train.head()
```

```
   ID                                          TITLE                                        ABSTRACT  Computer Science  Physics  Mathematics  Statistics  Quantitative Biology  Quantitative Finance
0   1        Reconstructing Subject-Specific Effect Maps  Predictive models allow subject-specific inf...                 1        0            0           0                     0                     0
1   2                 Rotation Invariance Neural Network  Rotation invariance and translation invarian...                 1        0            0           0                     0                     0
2   3  Spherical polyharmonics and Poisson kernels fo...  We introduce and develop the notion of spher...                 0        0            1           0                     0                     0
3   4  A finite element approximation for the stochas...  The stochastic Landau--Lifshitz--Gilbert (LL...                 0        0            1           0                     0                     0
4   5  Comparative study of Discrete Wavelet Transfor...  Fourier-transform infra-red (FTIR) spectra o...                 1        0            0           1                     0                     0
```

Image 1: Head of the train data

Barplot category :

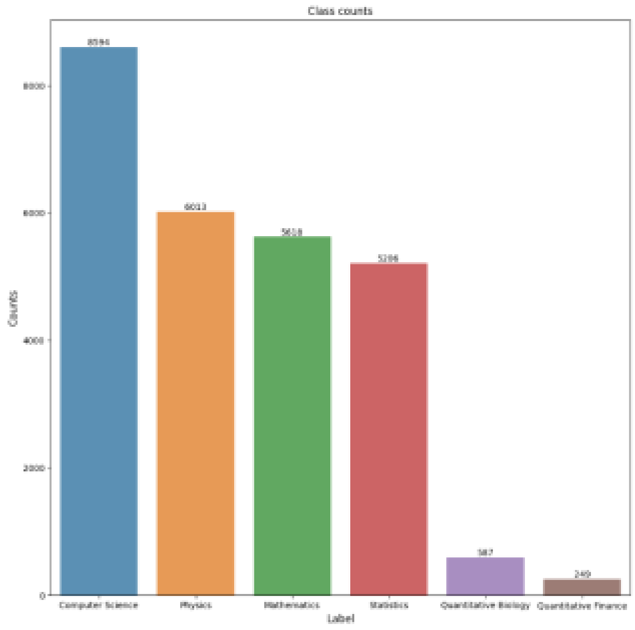


Image 2: Bar plot of the size of each class

Berdasarkan barplot diatas, dapat dilihat bahwa terdapat 8594 data dengan kategori “Computer Science”, 6013 data dengan kategori “Physics”, 5618 data dengan kategori “Mathematics”, 5206 data dengan kategori “Statistics”, 587 data dengan kategori “Quantitative Biology”, 249 data dengan kategori “Quantitative Finance”.

II. Methodology

A. Text Preprocessing

1. Lower

In text processing, the term "lower" refers to the operation of converting all alphabetic characters in a given text to lowercase. This process is used to achieve consistency and simplify various text-related tasks. By converting all characters to lowercase, regardless of their original case, we can eliminate the effects of case sensitivity when searching, comparing, or normalizing the text. It helps to ensure that words or phrases with the same letters but different cases are treated as identical. Lowercasing is particularly beneficial in natural language processing tasks, such as sentiment analysis or text classification, where the overall meaning and context of the text are more important than individual capitalization choices. By applying the lower operation, we create a standardized representation of the text, enabling easier and more accurate analysis and interpretation of the data.

2. **Removing Stopwords**
   In text processing, the removal of stop words refers to the process of eliminating common and insignificant words from a given text. Stop words typically include commonly used words such as "the," "a," "is," and "in," which do not contribute significantly to the overall meaning of the text. By removing these stop words, text processing tasks can focus on more meaningful and relevant words, thus improving the efficiency and accuracy of various natural language processing applications. Removing stop words can also help reduce the dimensionality of the text data, making it easier to analyze and extract meaningful patterns. However, the specific set of stop words to be removed may vary depending on the language and the specific task at hand.

3. **Remove Number, Punctuation, Special Characters, and Whitespace**
   In text processing, the removal of numbers, punctuation, special characters, and whitespace involves eliminating these elements from a given text to simplify and clean the data. Numbers, such as digits and numerical symbols, are typically irrelevant in many text analysis tasks and can be safely disregarded. Punctuation marks, such as periods, commas, and quotation marks, serve grammatical purposes but are often unnecessary for certain applications, such as keyword extraction or language modeling. Special characters, including symbols and non-alphanumeric characters, can introduce noise or inconsistencies in the text and are usually removed to ensure a more standardized and uniform dataset. Additionally, whitespace, such as spaces, tabs, and line breaks, is removed to condense the text and remove any unnecessary gaps. By performing these removal operations,

text processing becomes more efficient, and subsequent analyses can focus on the meaningful content of the text without being influenced by non-essential elements.

4. **Lemmatization/Stemming**
   Lemmatization and stemming are techniques used in text processing to simplify words by reducing them to their base or root forms. Lemmatization takes into account the context and part-of-speech of a word, transforming it into its dictionary or lemma form. This ensures that words with similar meanings are grouped together. For example, the lemma of "running" would be "run." On the other hand, stemming is a simpler process that chops off prefixes and suffixes to obtain a basic form, disregarding context. While stemming can lead to non-dictionary words, it is computationally less intensive than lemmatization. These techniques are employed to standardize words, aiding in tasks such as information retrieval and text analysis. Lemmatization provides more accurate results but requires more computational resources, while stemming offers a quicker but less precise approach.

5. **Tokenization**
   Tokenization is a fundamental process in text processing where a given text is divided into smaller units called tokens. These tokens can represent individual words, phrases, or characters, depending on the chosen approach. The main goal of tokenization is to break down the text into manageable and meaningful units that can be further analyzed. This process plays a vital role in various natural language processing tasks, such as text classification and sentiment analysis. By tokenizing the text, we can perform tasks like word frequency analysis, pattern identification, and statistical modeling. Tokenization also helps in handling punctuation and identifying sentence boundaries, contributing to a more structured and organized representation of the text. In summary, tokenization is a crucial step in text processing that allows for efficient and effective analysis and manipulation of text data.

B. Vectorization

The objective of vector space modeling is to map words within a language corpus onto a vector space, aiming to ensure that words with similar meanings are situated in proximity to one another (Krzeszewska et al.). There are several commonly used text vectorization methods for language processing namely Bag of Words, TF-IDF, and word embeddings.

For this project, we are going to use BERT tokenizer to vectorize our data. The BERT tokenizer can be seen as a vectorizer in the context of natural language processing tasks. While traditional vectorizers like TF-IDF and Bag-of-Words generate numerical representations based on word frequencies or occurrences, the BERT tokenizer transforms text into a sequence of tokens suitable for input into BERT models. Each token is assigned a unique identifier, and the tokenizer takes into account word boundaries, subword units, and special tokens like [CLS] and [SEP]. These tokens are then fed into the BERT model, which produces contextualized word embeddings. These embeddings capture the semantic meaning and contextual information of the text. Therefore, in the broader context of NLP, the BERT tokenizer acts as a vectorizer by transforming text into a sequence of tokens, which can be further processed by BERT models to obtain vector representations that capture the nuances of the text's meaning.

C. Modeling

Bidirectional Encoder Representations from Transformers (BERT) has been specifically created to pre-train extensive bidirectional representations from unannotated text, considering both the left and right context in all layers. Consequently, by incorporating a single additional output layer, the pre-trained BERT model can be fine-tuned to develop highly advanced models for various tasks, including question answering and language inference, without the need for significant adjustments to the task-specific architecture (Devlin et al.).

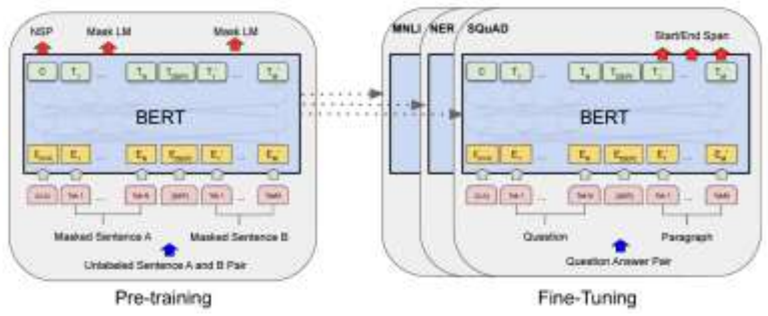


Image 3: BERT models for pre-training and fine-tuning

The BERT base model consists of an encoder with 12 Transformer blocks, each containing 12 self-attention heads, and a hidden size of 768. BERT accepts input sequences with a maximum length of 512 tokens and generates representations for the sequence. The sequence may have one or two segments, with the first token always being [CLS], which includes a special classification embedding. Another special token, [SEP], is used to separate segments.

For text classification tasks, BERT utilizes the final hidden state of the first token, [CLS], as the representation of the entire sequence. A simple softmax classifier is added on top of BERT to predict the probability of label c as

$$p(c|h) = softmax(Wh)$$

where W is the task-specific parameter matrix. To fine-tune BERT for the specific task, we optimize all the parameters from both BERT and W by maximizing the log-probability of the correct label (Chi et al.).

BERT is a suitable model for our multi-label classification problem for several important reasons:

1. Contextual Understanding: BERT captures contextual information by considering the surrounding words in both directions. This contextual understanding allows BERT to grasp the dependencies and relationships between words, which can be beneficial for multi-label classification tasks that require understanding the complex interactions between different labels.
2. Transfer Learning: BERT is pre-trained on a massive amount of unlabeled text from various domains, which helps it to learn general language representations. These pre-trained representations can be fine-tuned on specific datasets, including multi-label classification datasets, allowing BERT to leverage its pre-training knowledge and potentially achieve better performance.
3. Adaptive Representation: BERT's deep bidirectional architecture enables it to create rich and adaptive representations of words. It can capture both local and global dependencies, which can be advantageous for multi-label classification tasks that may involve long-range dependencies or nuanced relationships between labels.
4. One-to-Many Mapping: Multi-label classification involves assigning multiple labels to a single instance. BERT's ability to handle multiple outputs with a single additional output layer makes it convenient for multi-label classification without requiring substantial modifications to the model architecture.

Considering these factors, BERT emerged as the most suitable model for the multi-label data, providing a powerful combination of contextual understanding, transfer learning, adaptive representations, and support for multi-label classification.

III. Result and Analysis

By initializing the pretrained BERT model, the classification process can provide better results because the BERT pretrained model has been trained on a very large corpus of text, so it has a rich understanding of word meanings and the relationships between words. This model is capable of producing very informative word vector representations, which help in understanding the context of words in sentences. In addition, we use the adam optimizer with a learning rate of 2×10^{-5}.

Epoch	Train Loss	Train Accuracy
1	0.2263	0.7509
2	0.1589	0.7858
3	0.1303	0.8024
4	0.1043	0.8253
5	0.0803	0.8362

Tabel 1: Training loss and training accuracy for each epoch

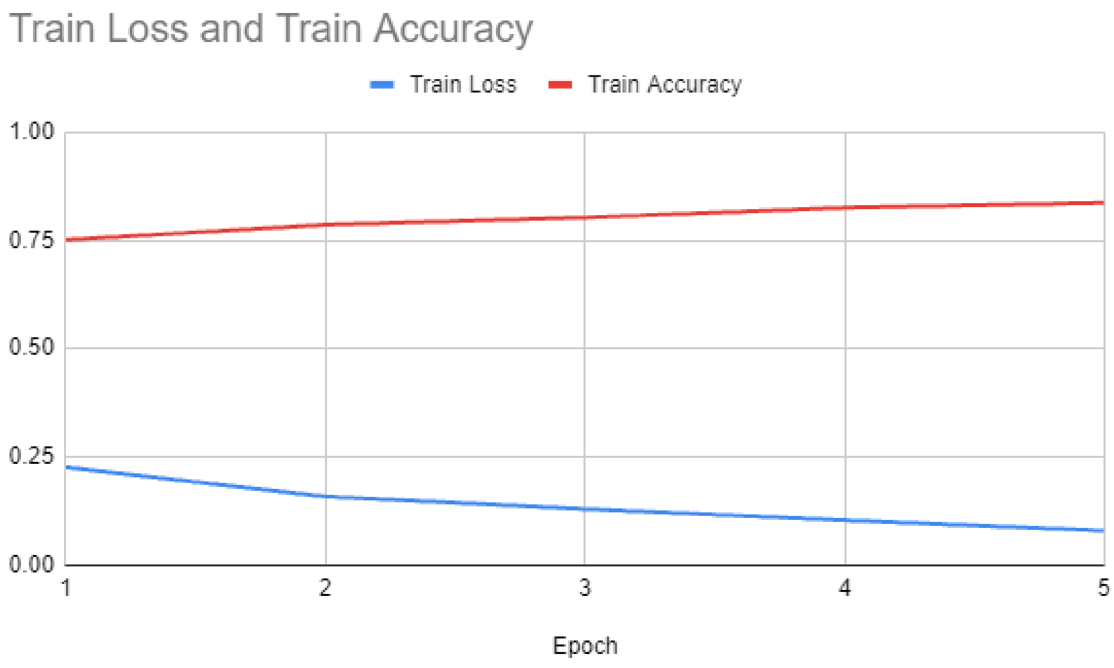


Image 4: Train Loss and Train Accuracy

This model is trained using 5 epochs. Based on the output of the training graph above, it can be seen that the training accuracy is increasing and the training loss is also decreasing with each epoch. After completing the training, we evaluate the model to predict the testing dataset. The following is the results of the classification report from the comparison of the predicted results of the dataset testing with the actual labels:

Confusion Matrix

	Computer Science	Physics	Mathematics	Statistics	Quantitative Biology	Quantitative Finance
Computer Science	1432	57	64	135	3	1
Physics	27	1060	25	7	11	0
Mathematics	88	56	710	41	2	1
Statistics	174	17	30	121	1	0
Quantitative Biology	17	21	0	15	39	0
Quantitative Finance	5	2	4	6	1	22

Table 2: Confusion Matrix

The confusion matrix is a tabular representation that is commonly used to evaluate the performance of a classification model. It provides a comprehensive summary of the predictions made by the model against the actual labels in the dataset.

Classification Report

	Precision	Recall	F1-score	Support
Computer Science	0.82	0.85	0.83	1692
Physics	0.86	0.92	0.89	1226
Mathematics	0.86	0.78	0.82	1150
Statistics	0.75	0.79	0.77	1069
Quantitative Biology	0.65	0.45	0.53	122

Quantitative Finance	0.84	0.70	0.70	45
Micro avg	0.82	0.83	0.82	5304
Macro avg	0.80	0.73	0.76	5304
Weighted avg	0.82	0.83	0.82	5304
Samples avg	0.86	0.86	0.84	5304

Table 3: Classification Report

Based on the table above, we can see that the Physics class has the highest metric score compared to the other classes. Physics has a precision of 0.86, which means that 86% of the Physics predictions made by the model are correct. Recall value is 0.92, which means the model can find 92% of all actual Physics cases. The F1 value is 0.89, which shows a good balance between the Precision and Recall values.

On the other hand, the Quantitative Biology class has the lowest performance score overall compared to the other classes with a precision score of 0.65 which indicates 65% of "Quantitative Biology" predictions are correctly classified. The recall score is merely 0.45 and in other words, the model only identified 45% of all actual "Quantitative Biology" cases. Moreover, the F1 score is 0.53 which gives an underwhelming result of both the precision and recall score. The considerable difference between the Qualitative Biology class performance when compared to the other classes can be explained by the significantly low number of observations (587) with that particular label.

In addition, we get the micro average result, which is 0.82, which means that the accuracy of this model reaches 82% overall if each sample is considered to have equal weight. For the macro average, the result is 0.80, which means that the accuracy of this model reaches 80% overall if each class is considered to have equal weight. Finally, for the weighted average, the result is 0.86, which means that the accuracy of this model reaches 86% overall if the number of samples in each class is different and is considered to affect the overall results.

IV. Conclusion

Based on the results of our research, by using the BERT model to perform multi-label classification on the NLP on Research Articles dataset, we conclude that this model provides fairly accurate predictions. With its ability to understand text context and generate rich representations, BERT can capture complex relationships between words and generalize well to multi-label classification tasks. Although there are some challenges in managing the size and complexity of the model, our results show that BERT is able to deal with these problems well. Thus, we can conclude that the BERT model is a good and effective choice for solving multi-label classification tasks on this NLP dataset.

V. References

Google Colab, URL:
https://colab.research.google.com/drive/1p3Ji8Hux4uG5IcK6E5N6vsiLsHj1jzEW?usp=sharing

Krzeszewska, U., Poniszewska-Maranda, A., Ochelska-Mierzejewska, J. 2022. *Systematic Comparison of Vectorization Methods in Classification Context.* URL: http://dx.doi.org/10.3390/app12105119

Devlin, J., Chang, M. W., Lee, K., Toutanova, K. 2018. *BERT:Pre-training of Deep Bidirectional Transformers for Language Understanding*. URL: https://doi.org/10.48550/arXiv.1810.04805

Sun, C., Qiu, Xu, Y., Huang, X. 2020. *How to Fine-Tune BERT for Text Classification?*. URL: https://doi.org/10.48550/arXiv.1905.05583

Mohan, Vijayarani. 2015. *Preprocessing Techniques for Text Mining - An Overview*. URL:
https://www.researchgate.net/publication/339529230_Preprocessing_Techniques_for_Text_Mining_-_An_Overview