Ma412 - Report

Multi-Label Classification of Scientific Literature Using the NASA SciX Corpus

Antoine DUMON
December 2024

Table des matières

1 Introduction

In the modern era, scientific research generates an ever-growing volume of publications across diverse fields. Effective management and organization of this wealth of information are critical for facilitating knowledge discovery and accelerating innovation. Traditional manual methods of categorizing scientific texts are no longer scalable, necessitating automated approaches to classify and index research documents efficiently.

This project explores multi-label classification techniques to predict relevant keywords for scientific articles. Using the NASA SciX dataset, which contains annotated abstracts and their corresponding keywords, we implement and evaluate three classification methods : Logistic Regression, Support Vector Machines (SVM), and Naive Bayes. Furthermore, we combine these methods to create a hybrid approach, aiming to improve prediction performance.

The focus of this study is not only to assess the individual strengths and limitations of these models but also to identify parameter configurations and hybrid strategies that optimize multi-label text classification for scientific documents.

2 Dataset and Metrics

In this section, we will see we need to know before looking for the best model for Multi-Label Classification. This will be mainly information about the data set that we are using and the metric that we will use to evaluate our models.

2.1 Dataset

The dataset is composed of 18 677 articles and for each of them we have five columns :
— bibcode : an id for each article
— title : the title of the article
— abstract : the abstract of the article
— verified_uat_ids : id for each label
— verified_uat_labels : keyword that are in the abstract/article

Some columns are not relevant such as bibcode and verified_uat_ids since there is no important information in thoose one. The two column on which we will focus our attention are "abstract" and "verified_uat_labels". We will use the abstract in order to find labels to categorize the article. Since there is already associated label for each article, this will allow us to use a machine learning algorythm which can be train on the dataset.
There is also the column title, some think that it is a good idea to merge title and abstract, so the algorythm have more word to base on. Personnaly, I don't think it is relevant and I also think that it can create bad result. If we merge the title and the abstract we will not add additionnal word but some existing label, it will just change the weight of each label. The one that are in the abstract and in the tiltle will weight more than the other and this can introduce bias in the model.

In total, we have 1864 labels. Some are more common than some other, the most used label are found in more than 900 articles. For the least common labels, we find them in only one article out of the 18 677 ones. The different weight of the label will be a problem because it is highly probable that the model will do error with the rarest label since he can't learn them correctly. We will have to take it into account when we will choose a way to evaluate the model and the metric system.

2.2 Metrics

2.2.1 Accuracy Score

It is a very intuitive metric that measure the proportion of correct predicted labels :

$$\text{Accurracy} = \frac{\text{Number of correct prediction}}{\text{Number of labels}}$$

We obtain a number between 0 and 1 with 1 meaning that all the predictions are good. This metric is not the most relevant but we will still use it in order to have a quick idea of the performance of the model. To compare models between them, we will most likely use F1-score.

2.2.2 F1-score

The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is :

$$F1 = \frac{2 * TP}{2 * TP + FN + FP}$$

Where
— TP is the number of true positives,
— FN is the number of false negatives, and
— FP is the number of false positives.
F1 is by default calculated as 0.0 when there are no true positives, false negatives, or false positives.

There is a difference between micro and macro F1-score. Macro will compute the F1-score for each label and find their unweighted mean. Micro will calculate metrics globally by counting the total true positives, false negatives and false positives.

We will use both of them, macro is good in order to have a global idea of the performance of the model, but since we know that some labels are very rare and that the model will probably be false on them we also like to have micro. The dataset is very

unbalanced with some label that are way more common and for this type of dataset micro is better.

So F1-score (micro) will be our main tool to compare models performances. The F1-score (macro) and the accuracy will also be display in order to have a general idea of the performance.

3 Model

3.1 Methods of vectorization

First of all we will compare different methods of vectorization (CountVectorizer and TFIDF) and with different parameter for each of them.

The **CountVectorizer** is a text vectorization method that converts a collection of text documents into a numerical representation by counting the occurrences of each unique word in the text. It tokenizes the text, builds a vocabulary of unique words, and represents each document as a sparse matrix where each row corresponds to a document, and each column corresponds to a word's count from the vocabulary. This method is simple and effective for capturing frequency-based text patterns but does not account for the semantic relationships between words.

The **TF-IDF** (Term Frequency-Inverse Document Frequency) is a numerical representation of text data that reflects the importance of a word in a document relative to a collection of documents (the corpus). It improves upon simple word frequency methods by assigning higher weights to words that are important to a document but occur less frequently across the corpus. TF-IDF is widely used in natural language processing tasks to convert textual data into features for machine learning models.

Term Frequency (TF) : Measures how often a term appears in a document, normalized to account for document length :

$$TF(t,d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

Inverse Document Frequency (IDF) : Reduces the weight of terms that appear frequently across many documents, as they are less informative :

$$IDF(t,D) = \log(\frac{\text{Total number of documents in corpus D}}{1 + \text{Number of documents containing term t}})$$

TF-IDF Score : Combines the two measure to assign a weight to each term in a document :

$$\text{TF-IDF}(t,d,D) = TF(t,d) \times IDF(t,D)$$

Model of vectorization

We will try six different model of vectorization :

- CountVectorizer (default) : With a max feature of 5000 word
- CountVectorizer (bigrams) : Similar to the default CountVectorizer, but considers both unigrams and bigrams (sequences of two consecutive words), which can capture more contextual information.
- TFIDFVectorizer (default) : With a max feature of 5000 word
- TFIDFVectorizer (bigrams) : A TF-IDF-based vectorizer that considers both unigrams and bigrams, adding more contextual understanding.
- TFIDFVectorizer (sublinear TF) : Similar to the default TF-IDF but applies logarithmic scaling to term frequencies, reducing the impact of high-frequency terms while preserving their importance.
- TFIDFVectorizer (unlimited features) : A TF-IDF vectorizer without a limit on the number of features (vocabulary size), capturing all unique words in the dataset for a more detailed representation.

All the vectorization method will be tested using the same method, here OneVsRestClassifier combined with LogisticRegression method.

Models	Accuracy	F1-score (micro)	F1-score (macro)
CountVectorizer (default)	0.015	0.3427	0.1461
CountVectorizer (bigrams)	0.0139	0.3462	0.1443
TFIDFVectorizer (default)	0.0051	0.3751	0.1796
TFIDFVectorizer (bigrams)	0.0043	0.3728	0.1788
TFIDFVectorizer (sublinear TF)	0.0075	0.4022	0.1787
TFIDFVectorizer (unlimited features)	0.0088	0.3921	0.1794

We can observe that the TF-IDF Method outperform clearly the CountVectorizer on the F1-score, The accuracy is better for the CountVectorize but as we said it is not the most relevant metric.

Among the different parameter that were tested we can say that they all have the same F1-score (macro). We will focus on the F1-score (micro) that is also the most significant.

The default and bigrams models are both similiar are getting outperform by the sublinear TF and the unlimited feature one. The sublinear model is a bit better than unlimited feature but there is an other factor that will confort us in the decision of choosing the sublinear one : the time. The sublinear TF is way more quicker to compute and have a better result so we will choose this model for the vectorization of the abstract.

This result could have been predicted, because the sublinear-TF method aim to reduce the impact of the common label without interfering with their importance. Since we have a very unbalanced dataset, this is a good model for us.

3.2 Methods of Classification

In this part, we will try different methods of classification with differents parameter for each of them : Logistic Regression, Support Vector Machine and Naive Bayes. All of them will be train using OneVsRestClassifier in order to handle Multi-Label Classification. The metric to judge the result is going to be the same that previously.

3.2.1 Logistic Regression

Logistic Regression is a linear model for binary classification, but we extend it to multi-label classification using the OneVsRestClassifier. It predicts the probability of each label and applies a threshold (e.g., 0.5) to assign labels to the data.The C parameter controls regularization strength, balancing underfitting and overfitting. Here we will try 3 different value of c (1, 0.5 and 2) :

Models	Accuracy	F1-score (micro)	F1-score (macro)
LogisticRegression (default)	0.0075	0.4022	0.1787
LogisticRegression (c = 0.5)	0.0062	0.3858	0.1778
LogisticRegression (c = 2)	0.0094	0.4133	0.1800

Logistic Regression is well-suited for this task, as evidenced by its consistently strong performance. Increasing the C value improves all metrics, with $C = 2.0$ achieving the highest F1-score (micro). A larger C reduces regularization, allowing the model to fit more closely to the data. This improvement aligns with the dataset's complexity, where lower regularization allows the model to capture finer patterns. However, excessive C could lead to overfitting, though it was not observed in this experiment.

3.2.2 Support Vector Machine

SVMs are linear classifiers that maximize the margin between data points of different classes. Using the OneVsRestClassifier, SVMs can handle multi-label problems. The regularization parameter C adjusts the trade-off between achieving a larger margin and minimizing classification errors.

Models	Accuracy	F1-score (micro)	F1-score (macro)
Support Vector Machine (default)	0.0209	0.2755	0.0637
Support Vector Machine (c = 0.5)	0.0225	0.3118	0.0758
Support Vector Machine (c = 2)	0.0195	0.2468	0.0549

SVM performance improves with $C = 0.5$, which balances the trade-off between maximizing the margin and minimizing classification errors. Increasing C to 2.0 results in overfitting, as the model focuses too heavily on correctly classifying training examples at the expense of generalization. This behavior explains the drop in F1-score (micro) when $C = 2.0$.

3.2.3 Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming independence between features. The MultinomialNB variant is specifically designed for discrete data like word counts or TF-IDF scores. The alpha parameter smooths probabilities, addressing zero-frequency issues.

Models	Accuracy	F1-score (micro)	F1-score (macro)
Naive Bayes (default)	0.0059	0.0479	0.0033
Naive Bayes (alpha = 0.5)	0.0091	0.1138	0.0089
Naive Bayes (alpha = 2)	0.0027	0.0207	0.0015

Naive Bayes performs poorly due to its simplifying assumption that features are conditionally independent given the label. This assumption is unrealistic for textual data, where word dependencies are common. The smoothing parameter
alpha slightly improves performance at
$alpha = 0.5$, as it helps address zero-frequency issues. However, larger
alpha values overly smooth probabilities, diluting the model's ability to discriminate effectively.

3.3 Hybrid Model

Now we will, create an hybrid model that will take the result of the three model and mix them together. We will use a logical "or" so when a model don't have any label we can complete with other models labels. We use the best performing parameter for each method so LogisticRegression (c = 2), Support Vector Machine (c = 0.5) and Naive Bayes (alpha = 0.5). We obtain this :

Metrics	Results
Accuracy	0.00847
F1-score (micro)	0.4133
F1-score (macro)	0.1800

We can observe that the result are the exact same than for the LogisticRegression (c=2) that we used in this hybrid model. It means that the other models predicts the same label. The three models are not complementary and produce the same result, LogisticRegression is just more performant.

4 Conclusion

This report presents an in-depth evaluation of three classification models—Logistic Regression, Support Vector Machines, and Naive Bayes—for multi-label text classification on scientific abstracts. Each model's performance was assessed using accuracy, micro F1-score, and macro F1-score, with varying parameter configurations.

Logistic Regression emerged as the most effective model, achieving the highest micro F1-score (0.4133) with a regularization parameter of $C = 2.0$. Its probabilistic nature and ability to handle high-dimensional data make it particularly well-suited for this task. SVM, while demonstrating competitive performance with $C = 0.5$, was sensitive to parameter adjustments and struggled with the inherent imbalance in the dataset. Naive Bayes performed the poorest, largely due to its simplifying assumption of feature independence, which fails to capture the complex relationships in textual data.

The hybrid model, combining the predictions of all three classifiers, showed no improvement over Logistic Regression alone. This suggests that the other models do not contribute additional complementary information, reaffirming Logistic Regression as the most robust choice for this dataset.

In summary, this study highlights the importance of model selection and parameter tuning in multi-label classification tasks. Future work could explore more advanced models, such as ensemble methods or deep learning architectures, to further enhance classification performance and scalability.