Text Classification of Medical Transcript

Lee Kah Win (P-COM0230/20)
CDS522, MSc Data Sc. & Analytic
School of Computer Sciences, USM
Penang, Malaysia
leekahwin84000@student.usm.my

***Abstract*— Medical domain is in a data rich environment that a variety of knowledge can be extracted for positive outcomes. This paper will show multiclass classification of medical transcript using a real dataset. The objective of this paper is to classify medical transcript based on the medical specialty labels, namely Discharge Summary, Neurosurgery and ENT. Text normalisation has been performed on the dataset and five different n-gram feature representations are extracted. Moreover, three supervised learning classifiers are trained on each of the n-gram feature representations, namely K-Nearest Neighbours, Decision Tree, and Random Forest. The classification performance is evaluated by the metric score of macro F1. The best score achieved is 0.93 macro F1 on testing set using tuned Random Forest and unigram feature vectors. Last but not least, some text normalisation tools are also deployed at my own website.**

Keywords—Text Classification, N-Gram, Random Forest, Decision Tree, K-Nearest Neighbours, Macro F1

I. Introduction

A. Overview

The overview of the paper is as followed. The motivation and n-gram are described in Section I, followed by problem and related works in Section II and III. Next, dataset will be described in Section IV meanwhile the proposed solutions will be enumerated in Section V. Furthermore, evaluation criteria will be made clear in Section VI and the discussion is written in Section VII. Lastly, this project ended with a conclusion in Section VIII and an Appendix.

B. Motivation

"Most of the knowledge in the world in the future is going to be extracted by machines and will reside in machines" [1, p.64].

The above quotes by Gutierrez in 2014 describes how important Artificial Intelligence in extracting data-driven knowledge in everyday people life. In like manner, medical is served in a data-rich environments where a variety of knowledge can be extracted. And medical issues are still a primary concern for many people life. However, one of the key challenges for medical analytics is to deal with big data volumes in the form of unstructured text. For example, clinical notes, electronic device, opinions from online health portal, social media, and many others. In this respect, natural language processing (NLP) has been increasingly considered as an enabling tool to improve medical care for competitive advantages. By using NLP technique such as N-gram, one can extract, explain, and get insights into the datasets and corpus in peculiar ways. Such insights can contribute positively in various medical operational aspects, including feature extraction and text classification.

C. N-gram

In this project, n-gram will be used for feature extraction for this classification task, which is important to know it in detail. N-gram is the first NLP approach, which introduced by Markov in 1913 [2]. An N-gram is an N-character slice of a longer string. The intuition of the n-gram model is that instead of computing a prediction based on entire corpus, one can approximate the prediction by only contiguous slices sequence of n words [3]. There are two common purposes of using n-gram, which are feature extraction and text generation.

To explain feature extraction using n-gram with a demonstration of the sentence, "The student is alone happily". The number of n-gram features can be calculated by $k - n + 1$, where k is the number of words. The result is a bag-of-n-grams model [4] for a classifier to train the linguistic algorithm. Table I shows the demonstration of different n-gram feature representations.

TABLE I. Demonstration of N-gram Feature Extraction

N-gram	N-gram Generated Sentence	Number of N-gram Features
Unigram (1-Gram)	"The", "student", "is", "alone", "happily"	5
Bigram (2-Gram)	"The student", "student is", "is alone", "alone happily"	4
Trigram (3-Gram)	"The student is", "student is alone", "is alone happily"	3
Quandrigram (4-Gram)	"The student is alone", "student is alone happily"	2

II. Problem

Medical transcription is a voice-recorded medical reports that are dictated by medical practitioners such as, for example, physicians and nurses . With the increasing amount of storing unstructured medical transcript information, the need for effective and timely retrieval of relevant information is growing. One method is to assign topic for each of the medical transcript. In a traditional approach, healthcare workers will spend a lot of time scanning through the transcripts with a view on identifying key issues. There are also transcripts with complex cases that lead to information overload, delays or missing information [5]. Not only without classified topic labels on medical transcripts will bring into a tedious task just only to lookup certain information, but also may affect patient's health. Take example as below, a patient with a medical transcript saying about history of surgical procedures of Neurosurgery that requires great attention, but because without a proper label as "neurosurgery", thus it might be mixed with other transcripts in a hospital and causing the medical procedures to be delayed. In this project, we will propose an NLP approach to classify 3 medical topic labels "Neurosurgery", "Discharge" and "ENT" by using 300 medical transcripts.

III. Related Work

Literature review has been conducted and discovered that the common use of n-gram is to do feature extraction from large corpus [4], [6], [7]. From the papers that have reviewed, the procedure of feature extraction is as follows. At first, the text to be pre-processed by, for example, stop words removal, stemming and tokenisation. It is followed by breaking the text into n-gram features, in which three of the papers have used unigram, bigram and trigram representation. Moreover, all the authors [4], [6], [7] have a similar consensus that the reason of using n-gram feature representation is allowing respective models to increase generalisation by capturing context of nearby words. In the work by [4], they have conducted text classification ICU clinical notes based on clinical diagnoses using n-gram features and SVM model. The paper used F-measure to evaluate the performance and identify that the bigram features can achieve best results. In addition, according to the work by Lu in 2018 [8], they have implemented topic identification of health-related messages in online health community using n-gram features. They have stated that sequences of n-gram length more than 3 were shown to be not useful and might decrease the performance. Hence, in this project, we will extract the features until 3-gram, i.e., trigram.

IV. Dataset

Medical transcription public dataset is used, which is sourced from Kaggle [9] and MT samples [10]. The dataset that used in this project consists of two attributes, namely 'Transcription' and 'Medical Specialty'. The attribute 'Transcription' is the predictor whereas the 'Medical Specialty' is the target label. Both of the attributes are categorical (nominal) data type. The dataset has 300 transcription instances, where 108 of them labelled as 'Discharge Summary', 94 labelled as 'Neurosurgery' and the rest of 98 labelled as 'ENT' (i.e., Otorhinolaryngology). The total word count of the 'Transcription' attribute is 139,368 words and the average transcription word count is 465 words. In addition, the Figure 1 below shows frequency counts of medical transcript classes.

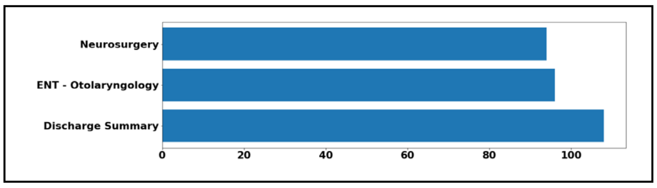
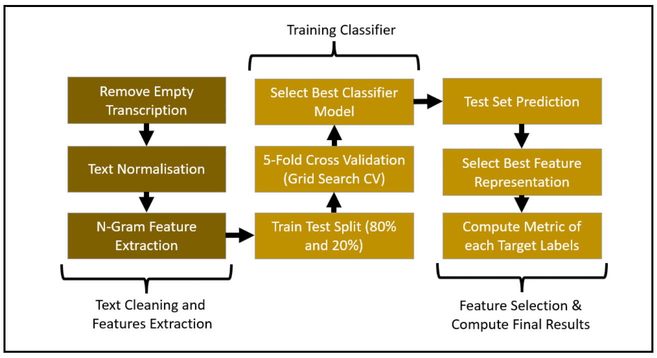


Fig. 1. Medical transcript classes value counts.

V. Proposed Solution

A. Solution Overview

First, the dataset will be pre-processed by text normalisation, which include: (1) lower cased, (2) remove punctuation and numbers, (3) tokenisation, (4) stemming and (5) stop words removal. Next, five types of n-gram feature vectors are extracted from the normalised dataset, i.e., (1) unigram, (2) unigram + bigram (3) bigram, (4) bigram + trigram and (5) trigram . Each of the n-gram feature vector are split randomly into 0.8 proportion train and 0.2 proportion test subsets. Then, 5-fold cross validation is applied on the training subset in order to find the optimised hyperparameters for the text classifier. After determining the best classifier, then the classifier is train on the 0.8 proportion dataset (i.e., training set). After training the model, the trained classifier is then use to predict the unseen data, which is the 0.2 proportion dataset (i.e., testing set). Macro F1 score metric will be used for evaluate the performance. The Figure 2 shows the solution overview flowchart.



Training Classifier
Remove Empty Transcription
Text Normalisation
N-Gram Feature Extraction
Select Best Classifier Model
5-Fold Cross Validation (Grid Search CV)
Train Test Split (80% and 20%)
Test Set Prediction
Select Best Feature Representation
Compute Metric of each Target Labels
Text Cleaning and Features Extraction
Feature Selection & Compute Final Results

Fig. 2. Solution overview flowchart.

B. Text Cleaning

Before conducting text cleaning, 2 empty transcriptions are identified. Hence, they are dropped as this attribute is the main predictor in this classification task, which cannot have null values. Text normalisation will be conducted for all of the transcriptions. The main reason of text normalisation is to convert the transcripts into standard format, which is important for feature extraction later. In this data normalisation task, the following task will be executed as depicted in Figure 3.

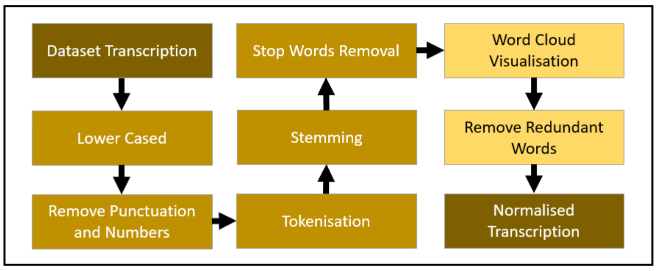
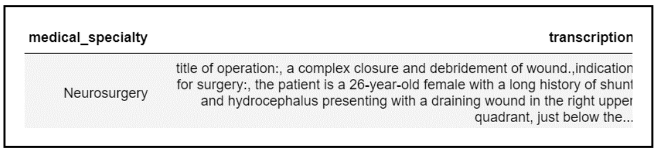


Fig. 3. Text normalisation flowchart.

In the following contents, text normalisation will be explained step by step.

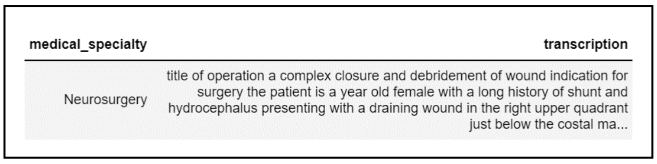
a) Lower Case: The transcriptions are in different capitalization, i.,e., lower case, capital case or upper case. This issue will lead to the dataset itself has many unique values as different casing of a word that a algorithm will treat it differently. Hence, all the transcriptions are lower cased. The figure below shows a transcript after lowering case.



medical_specialty	transcription
Neurosurgery	title of operation:, a complex closure and debridement of wound.,indication for surgery:, the patient is a 26-year-old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant, just below the...

Fig. 4. Transcript after lowering case.

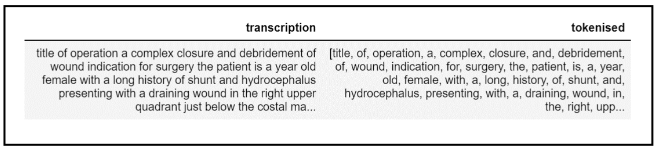
b) Remove Punctuation and Numbers: The unhelpful parts in the transcription are removed, which are, in this project, punctuation and numbers. Punctuations are removed by using the regular expression rule of '[\w]+'. Meanwhile numbers are removed by matching all numbers using the regular expression rule of '\d+' and then delete it (i.e., replacing it with nothing). The figure below shows the transcript after removing punctuation and numbers.



medical_specialty	transcription
Neurosurgery	title of operation a complex closure and debridement of wound indication for surgery the patient is a year old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant just below the costal ma...

Fig. 5. Transcript after removing punctuation and numbers.

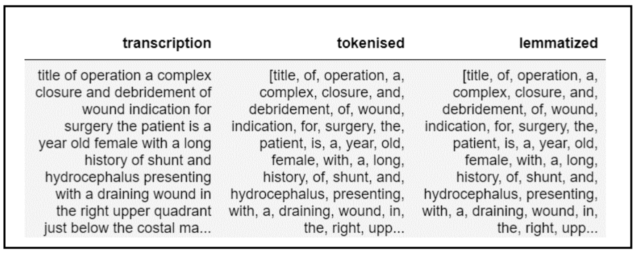
c) Tokenisation: Each transcript is then divided into lists of substrings based on whitespace by using 'NLTK WhitespaceTokenizer' Package. By doing so, not only additional whitespaces, tabs, new line are removed, but also to put the tokenised text into Stemming and Stop Words Removal procedures later on. The figure below shows the transcript after after tokenisation.



transcription	tokenised
title of operation a complex closure and debridement of wound indication for surgery the patient is a year old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant just below the costal ma...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...

Fig. 6. Transcript after tokenisation.

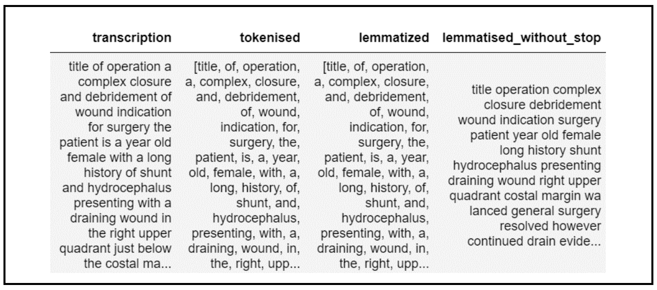
d) Lemmatisation: Each word in a tokenised transcript is then transformed into a base word by removing the suffix using 'WordNetLemmatizer'. Lemmatisation is chosen rather than Stemming due to I discovered that the Lemmatisation has a better performance in the multiclass classification task compared to stemming. The justification will be further discussed in Discussion Section VII. The figure below shows the transcript after lemmatisation.



transcription	tokenised	lemmatized
title of operation a complex closure and debridement of wound indication for surgery the patient is a year old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant just below the costal ma...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...

Fig. 7. Transcript after lemmatisation.

e) Stop Words Removal: There are 179 stop words to be removed for each transcript such as, for example, 'i', 'me' and 'should'. Those words are identified to be less informative, and they are frequently appeared in text causing adverse effect to a classifier performance as it distracted away the focus of important information. Hence, those stop words are removed from all the transcripts. The figure below shows the transcript after removing stop words.



transcription	tokenised	lemmatized	lemmatised_without_stop
title of operation a complex closure and debridement of wound indication for surgery the patient is a year old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant just below the costal ma...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...	title operation complex closure debridement wound indication surgery patient year old female long history shunt hydrocephalus presenting draining wound right upper quadrant costal margin wa lanced general surgery resolved however continued drain evide...

Fig. 8. Transcript after removing stop words.

After text normalisation, word clouds are generated to observe the importance of each word for the whole transcription and each subclass of them. Figures below show the word clouds.

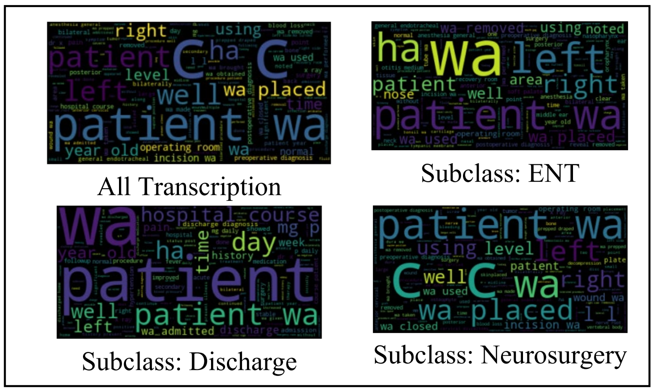


Fig. 9. Word clouds of medical transcript.

From generating the word clouds, some other frequently occurring words that have less semantic information are identified. Those words are "patient", "use", "wa", "ha" and "c". Those words are identified to be frequently occurred in each subclass as shown in Figure 9. Hence, they are removed from the medical transcripts. Figure below shows the word cloud after removing the redundant words.

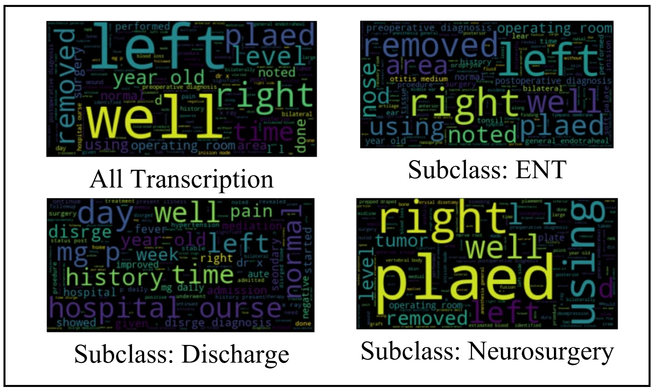


Fig. 10. Word clouds of medical transcript after removing redundant words.

After text cleaning, the total word counts are computed to show the word count differences before and after text cleaning. The results are summarised in Table II below.

TABLE II. WORD COUNT OF MEDICAL TRANSCRIPT

Data	Word Count		Change
	Before Cleaning	*After Cleaning*	
All Transcription	139368	79575	-42.90%
Subclass: ENT	42032	25290	-41.33%
Subclass: Discharge	43103	24242	-42.32%
Subclass: Neurosurgery	54233	30043	-57.55%

C. Feature Extraction

In this part, 5 different types of n-gram features will be extracted, which are (1) unigram, (2) unigram + bigram (i.e., a mixture of unigram and bigram features), (3) bigram, (4) bigram + trigram and (5) trigram. The first step is to convert the transcript column into a list, e.g., ['first row transcription', ‘second row transcription’, …]. Next, Scikit Learn library ‘CountVectorizer’ is used to define the ‘ngram_range’ in order to extract specific n-gram features. For example, ‘ngram_range’ of (1, 1) means only unigrams, (1, 2) means unigrams and bigrams, and (2, 2) means only bigrams. After that, iteratively converting the list of transcripts into a matrix of tokens based on the specified n-gram range. The vectorised n-gram features are extracted and the shape dimension of the vectors are shown as Table III below.

TABLE III. ARRAY DIMENSION OF N-GRAM FEATURES

Feature Vectors	**Lengths of array dimension**	
	Rows	***Columns***
Unigram	298	7038
Unigram + Bigram	298	56270
Bigram	298	49232
Bigram + Trigram	298	114215
Trigram	298	64983

It is interesting to notice that when the number of 'n' getting higher (i.e., n=1:unigram, n=2:bigram, n=3:trigram), there is more columns. But actually, it has less features to be generated for each transcript. One may wonder why this happens? This is due to it is getting harder to find similar features that can be stored in similar column when it has a longer connected words as one feature. If the feature is unique, it will automatically append additional column to store the feature. Nevertheless, the higher the number of ‘n’ will contribute to the features becoming more context sensitive to the transcription as considering more adjacent words. Meanwhile it also contributes to the algorithm become more complex as learning more features. A score metric comparison will be conducted later to identify which n-gram features can has a best performance.

D. Topic Classification

Three classification machine learning models are used in this multiclassification task, namely K-Nearest Neighbour (KNN), Decision Tree Classifier (DTC) and Random Forest Classifier (RFC). They are selected to be trained on the feature vectors in order to know which model can generate best performance. The reasons of selecting these three classification models will be further discussed in Discussion Section.

Now let’s discuss each of the classifier. KNN is a distance-based classifier that can learn the data to separate the text based on their similarity with neighbours. In addition, DTC is capable to learn the decision rules inferred from the data features and it able to handle multi-output problems, just like in this project. However, it has overfitting issue as it creates over-complex trees that do not generalise the data well. This issue has brought into ensemble learning method. The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability. One popular ensemble method is RFC, each decision tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set [11]. Random forests achieve a reduced variance by combining diverse trees allowing to overcome the overfitting problem brought by DTC [12].

In order to find the best parameters for each of the classifier, GridSearchCV (or more specifically HalvingGridSearchCV, see Appendix for more information) is used that it enables an exhaustive search over specified parameter values for an estimator. Feature vector of unigram is used for the GridSearchCV as I discovered in later process that unigram feature vector has the best performance compared to other feature vectors. It is important to point out that learning the parameters of a prediction function and testing it on the same data is a methodological mistake. This is because a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on unseen data. This situation is called overfitting [11]. Hence, the procedures of this classification task are defined and utilised, which as shown in the flow chart below.

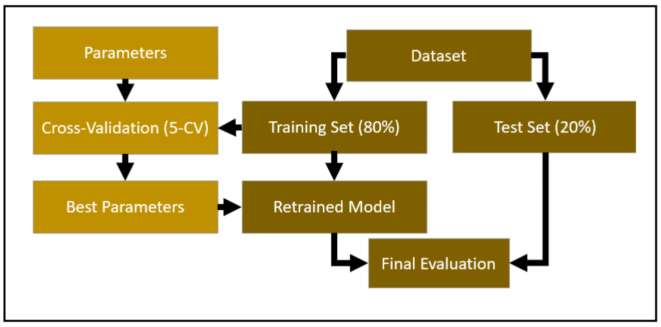


Fig. 11. Building text classifier flow chart.

First, the dataset is split into 80% for training and 20% for testing set. In order to solve overfitting issue, cross-validation (CV) is applied on the training data. In this project, 5-fold CV is implemented, rather than 10-fold as considering the dataset size is not very large, i.e., 298 transcription instances. The 5-fold CV enables the training set to split into 5 smaller sets as shown in Figure 12. Then, each model is trained using the folds as training data and the resulting model is validated on the remaining part of the data (i.e., testing data).

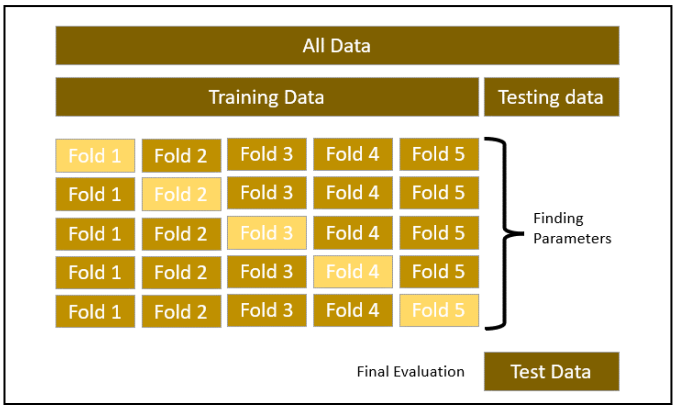


Fig. 12. 5-fold cross validation.

In this 5-fold cross validation setting, GridSearchCV is used to find the optimised hyperparameters in this process. The metric used for the GridSearchCV in the 5-fold CV is macro F1, which is preferable for multiclass classification task. The additional justifications of choosing macro F1 will be discussed in next section. After the exhaustive search, the results of the optimised parameters and its best score are obtained, which tabulated as in Table IV below.

TABLE IV. BEST PARAMETERS OF EACH SELECTED CLASSIFIER

Classifier	Best Parameters	Best macro F1
KNN	n_neighbors = 7	0.63
DTC	max_depth = None, min_samples_split = 2	0.62
RFC	max_depth = 35, n_estimators = 16	0.87

VI. Evaluation Criteria

The evaluation metric that used here is macro F1. In order to know what is macro F1, one should know what is F1 score. The F1 score is the harmonic mean of the precision and recall [11], where an F1 score reaches its best value at 1 and worst score at 0. The formula (1) is as shown below.

$$(precision * recall) / (precision + recall) = F1 \quad (1)$$

For the case of macro F1, it computes the average of the F1 score of each target class. The main reason of using macro F1 is because this project is a multiclass classification so that macro F1 can assess the average metric for three of the class labels. Moreover, macro F1 combines the precision and recall of a classifier into a single metric by taking their harmonic mean and average of each targe class. Hence, macro F1 is chosen to evaluate the classification performance fairly.

In this section, let's discuss how to evaluate the performance after identifying the best classifier. The result shows that the RFC with the hyperparameters of "max_depth = 35, n_estimators = 16" can achieve the best result in cross validation setting with a macro F1 of 0.87. But how is the performance on the data that is not seen? To evaluate this issue, the RFC with the best hyperparameters is fitted into the 80% training dataset to learn the algorithm. Then, the trained RFC is then used to predict on the unseen data (i.e., 20% testing set).

The metric score of predicting the unseen data is summarised as in Table V below. We noticed that the unigram features have the best macro F1 score at 0.9336.

TABLE V. Results of Classification Metric Score

Features	macro F1
Unigram	0.9336
Unigram + Bigram	0.8526
Bigram	0.8372
Bigram + Trigram	0.8499
Trigram	0.5773

The unigram feature vector is then used to compute for each target class metric score. One may wonder why macro F1 is not shown in the column, this is because macro F1 only works when it involved multiple classes. Table VI shows the result of each target class metric score.

TABLE VI. Results of Each Target Class Metric Score

Target Labels	Precision	Recall	F1	Support
Discharge Summary	0.94	0.94	0.94	18
ENT	1.00	0.88	0.93	24
Neurosurgery	0.86	1.00	0.92	18

VII. Discussion

In this section, several issues will be discussed, which are reasons of using lemmatisation, machine learning models, performance and feature vectors.

A. *Reasons of Using Lemmatisation*

Lemmatisation is used here not only helps to achieve the root forms of each word without grammatically error like stemming does, but the algorithm performance use lemmatisation is better than stemming. To justify that, we used back the best feature vector as discussed previously (i.e., unigram) and GridSearchCV with a 5-fold cross validation is used to find the best macro F1 for both of the steamed and lemmatised feature representation. All other experimental settings are fixed in order to make an apple-to-apple comparison. The results are obtained and as shown in Table VII. The results show that the lemmatised dataset is slightly better in this classification task, hence the lemmatisation is used as we described in the text normalisation section.

TABLE VII. Performance Comparison of Lemmatisation and Stemming

Dataset	macro F1	Parameters
Stemmed	0.86	RandomForestClassifier(max_depth= 35, n_estimators=16, random_state=8888)
Lemmatised	0.87	

B. *Machine Learning Models of KNN, DTC and RFC*

Three of the machine learning classifiers (i.e., KNN, DTC and RFC) are selected because they are nonparametric models, which are suitable in this multiclass classification task. To describe, these are the algorithms do not make strong assumption about the data and has a few parameters can be tuned using, as mentioned, GridSearchCV. To describe, KNN can tune, for example, number of neighbours. Meanwhile DTC and RFC can tune the maximum depth of the tree and minimum number of samples required to split an internal node. Hence, these algorithms are free to learn any functional form about the data. Moreover, the dataset consists a lot of features, i.e., at least 7038 and above. If using parametric model like Naïve Bayes, which makes strong statistical assumption, is unlikely to map the true underlying function of the data. So, by using these three non-parametric classifiers in this multiclass classification task, it can result in a higher performance compared to parametric models.

C. *Performance*

The best performance of predicting the unseen data achieve 0.93 macro F1. The algorithm is using RFC with parameters of (max_depth=35, n_estimators=16) on a feature vector of unigram. The classifier is not underfitting as, not only it is a nonparametric model that does not make predictive assumption, but also it has a high macro F1 score. In another word, it has a high harmonic mean of precision and recall for each target class. Moreover, it is interesting to point out that how to confirm that the model is not overfitting? Can this performance metric score occurs based on pure luck? The actions to be validated further are running more random splits of training and testing dataset and use the trained model to predict different set of unseen data. The results are as shown in Table VIII below.

TABLE VIII. Metric Score on Different Train Test Split Random State

Train Test Split Random State	macro F1
Random State = 8888 [a]	0.9336
Random State = 3213	0.8977
Random State = 6747	0.8567
Random State = 4321	0.8626

[a] Default random state used in this Project

The results shown that the performance metric is not based on pure luck due to it is not significantly worse than the cross-validation metric score, i.e., 0.87 macro f1. Hence, the RFC model that is not overfitting in this context is deduced.

D. Feature Vector

In addition, let's talk about the feature vector. As the results shown, unigram feature vector can generate best performance, which indicate that the sequence of word in this classification context is less important. Unigram feature vectors show the probability of each word is independent of any words before it. In other words, training the model is nothing but calculating these fractions for all unigrams in the training dataset. As the number of "n" in n-gram increases, the metric score drops. The second best performed unigram + bigram feature vector (macro F1 = 0.85) and followed by bigram (macro F1 = 0.84). They are not performed poorly, instead they have a quite good performance in this multiclass classification task. However, unigram feature vector has a better performance than all of other feature representations. This result also shows that the individual keyword appearance in this corpus is a more important predictor rather than the combination of adjacent words. In fact, considering adjacent word sequence, i.e., neighbouring words, is one of the limitations of n-gram feature representation. Hence, we can deduce that the word ordering for the medical transcript in this context is less important in predicting different medical specialty topics.

VIII. Conclusion

In this project, text classification for medical transcripts has successfully implemented. Several NLP techniques have been used to get this project finished, such as text normalisation, feature extraction and word cloud. The best result of the prediction achieves 0.93 macro F1 on the testing subset using RFC and unigram feature representation. One key finding is that the adjacent word sequence does not improve the performance score in this context as the word ordering of medical transcript seems like less important in predicting the target classes. In future study, one can use skip-gram feature vector to classify the medical transcript in order to handle constituency variation (i.e., allow token to be skipped) [13]. I have gained a lot of applicable useful knowledge in natural processing. Furthermore, I have applied this text processing knowledge at my own profile by building a text normalisation tool in my website.

References

[1] Gutierrez, *Data scientists at work.* Apress, 2014.

[2] A. Markov, "Example of a statistical investigation of the text of 'Eugene Onegin' illustrating the dependence between samples in chain," *Izvistia Imperatorskoi Akademii Nauk (Bulletin de l'Academie Imp ′ eriale ′ des Sciences de St.-Petersbourg)*, pp. 153–162, 1913.

[3] D. Jurasfky and J. Martin, *An introduction to natural language processing, computational linguistics, and speech recognition.*, 3rd ed. 2021.

[4] B. J. Marafino, J. M. Davies, N. S. Bardach, M. L. Dean, and R. A. Dudley, "N-gram support vector machines for scalable procedure and diagnosis classification, with applications to clinical free text data from the intensive care unit," *Journal of the American Medical Informatics Association*, vol. 21, no. 5, pp. 871–875, 2014, doi: 10.1136/amiajnl-2014-002694.

[5] M. Hughes, I. Li, S. Kotoulas, and T. Suzumura, "Medical text classification using convolutional neural networks," in *In Informatics for Health: Connected Citizen-Led Wellness and Population Health*, 2017, pp. 246–250.

[6] J. Ashok Kumar, S. Abirami, and T. E. Trueman, "An N-Gram Feature-Based Sentiment Classification Model for Drug User Reviews," pp. 277–297, 2021, doi: 10.1007/978-981-16-2674-6_22.

[7] M. D. P. Salas-Zárate, J. Medina-Moreira, K. Lagos-Ortiz, H. Luna-Aveiga, M. Á. Rodríguez-García, and R. Valencia-García, "Sentiment Analysis on Tweets about Diabetes: An Aspect-Level Approach," *Computational and Mathematical Methods in Medicine*, vol. 2017, 2017, doi: 10.1155/2017/5140631.

[8] Y. Lu, "Automatic topic identification of health-related messages in online health community using text classification," *SpringerPlus*, vol. 2, no. 1, 2013, doi: 10.1186/2193-1801-2-309.

[9] T. Boyle, "Medical Transcriptions | Kaggle," *Kaggle.* https://www.kaggle.com/tboyle10/medicaltranscriptions (accessed Jan. 14, 2022).

[10] "Transcribed Medical Transcription Sample Reports and Examples - MTSamples," *MTSamples.* https://mtsamples.com/index.asp (accessed Jan. 14, 2022).

[11] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *The Journal of machine Learning research*, vol. 11, no. 1, Nov. 2011.

[12] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *International Journal of Computer Science Issues (IJCSI)*, vol. 1, no. 9, Sep. 2012.

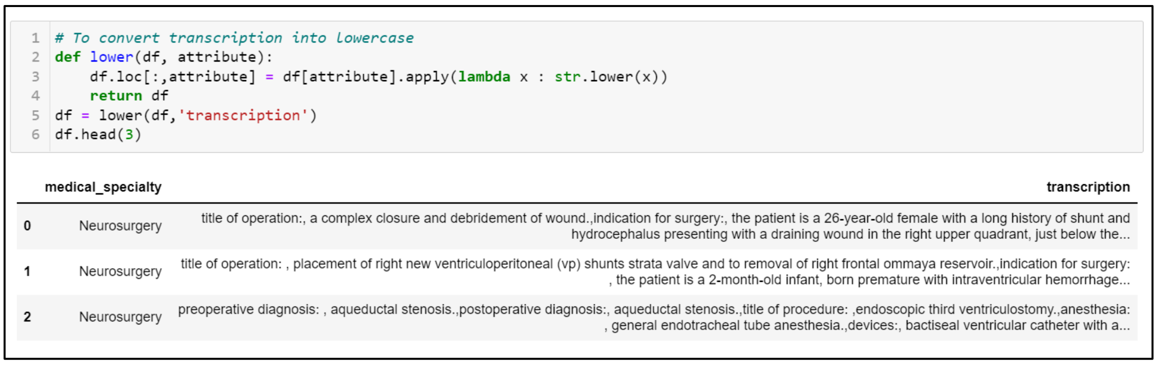
[13] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks, "A Closer Look at Skip-gram Modelling," *LREC*, vol. 6, pp. 1222–1225, 2006.

Appendix

1.0 Text Normalisation

1.1 To convert text to lower case

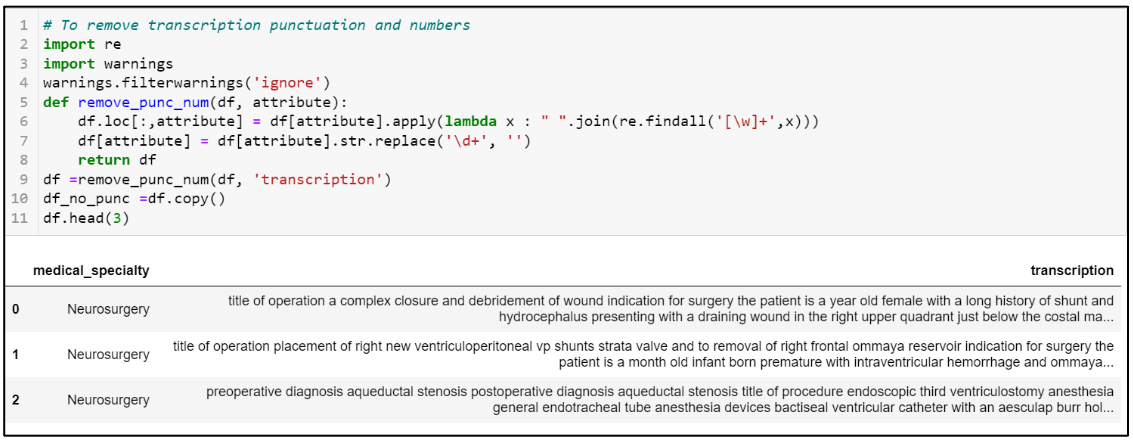
```
# To convert transcription into lowercase
def lower(df, attribute):
    df.loc[:,attribute] = df[attribute].apply(lambda x : str.lower(x))
    return df
df = lower(df,'transcription')
df.head(3)
```


```
# To convert transcription into lowercase
def lower(df, attribute):
    df.loc[:,attribute] = df[attribute].apply(lambda x : str.lower(x))
    return df
df = lower(df,'transcription')
df.head(3)
```

	medical_specialty	transcription
0	Neurosurgery	title of operation:, a complex closure and debridement of wound.,indication for surgery:, the patient is a 26-year-old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant, just below the...
1	Neurosurgery	title of operation: , placement of right new ventriculoperitoneal (vp) shunts strata valve and to removal of right frontal ommaya reservoir.,indication for surgery: , the patient is a 2-month-old infant, born premature with intraventricular hemorrhage...
2	Neurosurgery	preoperative diagnosis: , aqueductal stenosis.,postoperative diagnosis:, aqueductal stenosis.,title of procedure: ,endoscopic third ventriculostomy.,anesthesia: , general endotracheal tube anesthesia.,devices:, bactiseal ventricular catheter with a...

1.2 Removing Punctuation and Numbers

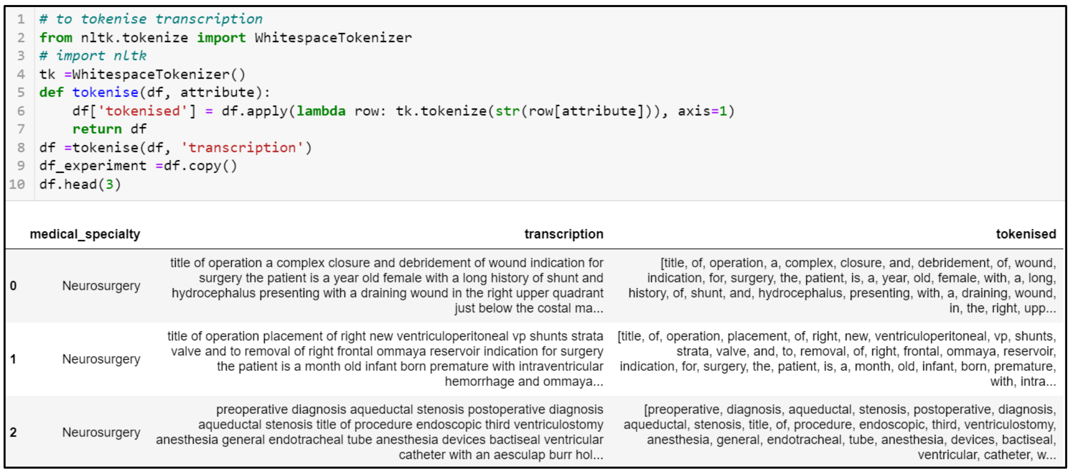
```
# To remove transcription punctuation and numbers
import re
def remove_punc_num(df, attribute):
    df.loc[:,attribute] = df[attribute].apply(lambda x : " ".join(re.findall('[\w]+',x)))
    df[attribute] = df[attribute].str.replace('\d+', '')
    return df
df =remove_punc_num(df, 'transcription')
df_no_punc =df.copy()
df.head(3)
```


```
# To remove transcription punctuation and numbers
import re
import warnings
warnings.filterwarnings('ignore')
def remove_punc_num(df, attribute):
    df.loc[:,attribute] = df[attribute].apply(lambda x : " ".join(re.findall('[\w]+',x)))
    df[attribute] = df[attribute].str.replace('\d+', '')
    return df
df =remove_punc_num(df, 'transcription')
df_no_punc =df.copy()
df.head(3)
```

	medical_specialty	transcription
0	Neurosurgery	title of operation a complex closure and debridement of wound indication for surgery the patient is a year old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant just below the costal ma...
1	Neurosurgery	title of operation placement of right new ventriculoperitoneal vp shunts strata valve and to removal of right frontal ommaya reservoir indication for surgery the patient is a month old infant born premature with intraventricular hemorrhage and ommaya...
2	Neurosurgery	preoperative diagnosis aqueductal stenosis postoperative diagnosis aqueductal stenosis title of procedure endoscopic third ventriculostomy anesthesia general endotracheal tube anesthesia devices bactiseal ventricular catheter with an aesculap burr hol...

1.3 Tokenisation

```
# to tokenise transcription
from nltk.tokenize import WhitespaceTokenizer
# import nltk
tk =WhitespaceTokenizer()
def tokenise(df, attribute):
    df['tokenised'] = df.apply(lambda row: tk.tokenize(str(row[attribute])), axis=1)
    return df
df =tokenise(df, 'transcription')
df_experiment =df.copy()
df.head(3)
```

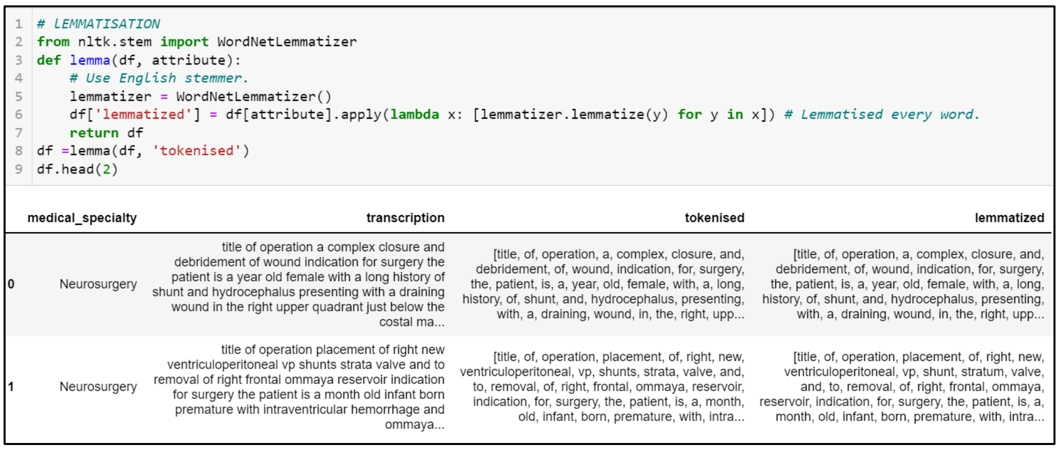


```python
# to tokenise transcription
from nltk.tokenize import WhitespaceTokenizer
# import nltk
tk =WhitespaceTokenizer()
def tokenise(df, attribute):
    df['tokenised'] = df.apply(lambda row: tk.tokenize(str(row[attribute])), axis=1)
    return df
df =tokenise(df, 'transcription')
df_experiment =df.copy()
df.head(3)
```

	medical_specialty	transcription	tokenised
0	Neurosurgery	title of operation a complex closure and debridement of wound indication for surgery the patient is a year old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant just below the costal ma...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...
1	Neurosurgery	title of operation placement of right new ventriculoperitoneal vp shunts strata valve and to removal of right frontal ommaya reservoir indication for surgery the patient is a month old infant born premature with intraventricular hemorrhage and ommaya...	[title, of, operation, placement, of, right, new, ventriculoperitoneal, vp, shunts, strata, valve, and, to, removal, of, right, frontal, ommaya, reservoir, indication, for, surgery, the, patient, is, a, month, old, infant, born, premature, with, intra...
2	Neurosurgery	preoperative diagnosis aqueductal stenosis postoperative diagnosis aqueductal stenosis title of procedure endoscopic third ventriculostomy anesthesia general endotracheal tube anesthesia devices bactiseal ventricular catheter with an aesculap burr hol...	[preoperative, diagnosis, aqueductal, stenosis, postoperative, diagnosis, aqueductal, stenosis, title, of, procedure, endoscopic, third, ventriculostomy, anesthesia, general, endotracheal, tube, anesthesia, devices, bactiseal, ventricular, catheter, w...

1.4 Lemmatisation

```python
# lEMMATISATION
from nltk.stem import WordNetLemmatizer
def lemma(df, attribute):
    # Use English stemmer.
    lemmatizer = WordNetLemmatizer()
    df['lemmatized'] = df[attribute].apply(lambda x: [lemmatizer.lemmatize(y) for y in x]) # Lemmatised every word.
    return df
df =lemma(df, 'tokenised')
df.head(2)
```

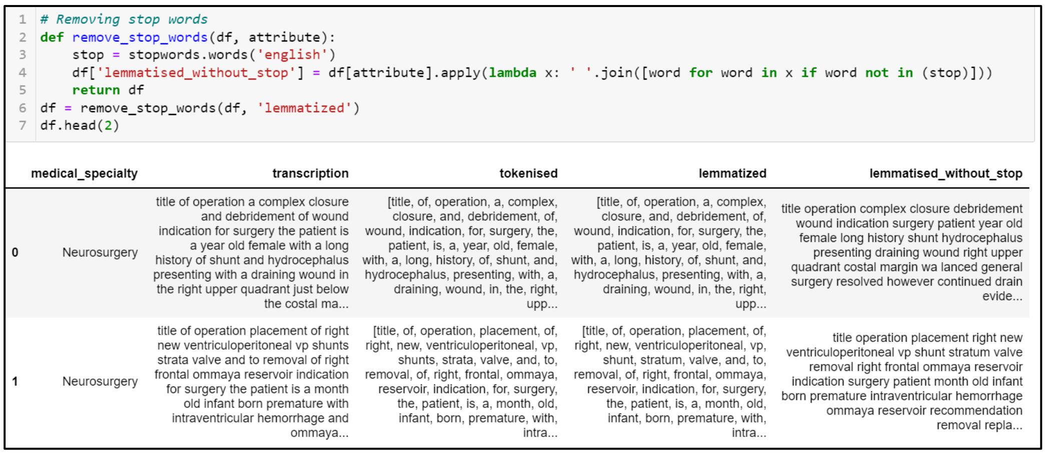


```python
# LEMMATISATION
from nltk.stem import WordNetLemmatizer
def lemma(df, attribute):
    # Use English stemmer.
    lemmatizer = WordNetLemmatizer()
    df['lemmatized'] = df[attribute].apply(lambda x: [lemmatizer.lemmatize(y) for y in x]) # Lemmatised every word.
    return df
df =lemma(df, 'tokenised')
df.head(2)
```

	medical_specialty	transcription	tokenised	lemmatized
0	Neurosurgery	title of operation a complex closure and debridement of wound indication for surgery the patient is a year old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant just below the costal ma...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...
1	Neurosurgery	title of operation placement of right new ventriculoperitoneal vp shunts strata valve and to removal of right frontal ommaya reservoir indication for surgery the patient is a month old infant born premature with intraventricular hemorrhage and ommaya...	[title, of, operation, placement, of, right, new, ventriculoperitoneal, vp, shunts, strata, valve, and, to, removal, of, right, frontal, ommaya, reservoir, indication, for, surgery, the, patient, is, a, month, old, infant, born, premature, with, intra...	[title, of, operation, placement, of, right, new, ventriculoperitoneal, vp, shunt, stratum, valve, and, to, removal, of, right, frontal, ommaya, reservoir, indication, for, surgery, the, patient, is, a, month, old, infant, born, premature, with, intra...

1.5 Stop Words Removal

```python
# Removing stop words
def remove_stop_words(df, attribute):
    stop = stopwords.words('english')
    df['lemmatised_without_stop'] = df[attribute].apply(lambda x: ' '.join([word for word in x if word not in
(stop)]))
    return df
df = remove_stop_words(df, 'lemmatized')
df.head(2)
```

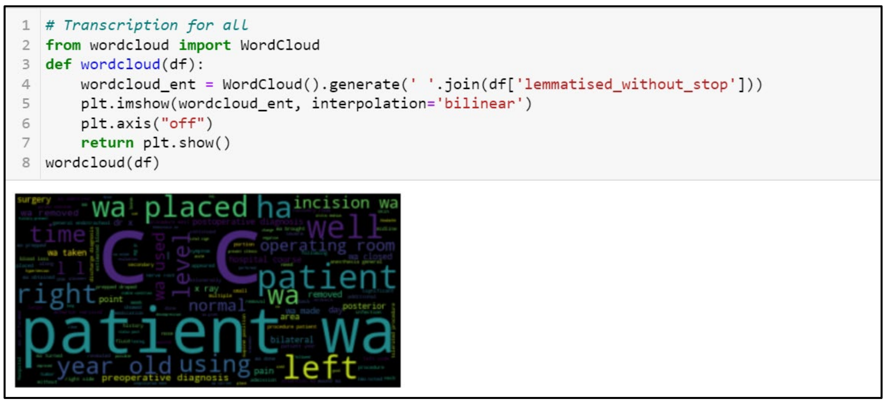


```
# Removing stop words
def remove_stop_words(df, attribute):
    stop = stopwords.words('english')
    df['lemmatised_without_stop'] = df[attribute].apply(lambda x: ' '.join([word for word in x if word not in (stop)]))
    return df
df = remove_stop_words(df, 'lemmatized')
df.head(2)
```

	medical_specialty	transcription	tokenised	lemmatized	lemmatised_without_stop
0	Neurosurgery	title of operation a complex closure and debridement of wound indication for surgery the patient is a year old female with a long history of shunt and hydrocephalus presenting with a draining wound in the right upper quadrant just below the costal ma...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...	[title, of, operation, a, complex, closure, and, debridement, of, wound, indication, for, surgery, the, patient, is, a, year, old, female, with, a, long, history, of, shunt, and, hydrocephalus, presenting, with, a, draining, wound, in, the, right, upp...	title operation complex closure debridement wound indication surgery patient year old female long history shunt hydrocephalus presenting draining wound right upper quadrant costal margin wa lanced general surgery resolved however continued drain evide...
1	Neurosurgery	title of operation placement of right new ventriculoperitoneal vp shunts strata valve and to removal of right frontal ommaya reservoir indication for surgery the patient is a month old infant born premature with intraventricular hemorrhage and ommaya...	[title, of, operation, placement, of, right, new, ventriculoperitoneal, vp, shunts, strata, valve, and, to, removal, of, right, frontal, ommaya, reservoir, indication, for, surgery, the, patient, is, a, month, old, infant, born, premature, with, intra...	[title, of, operation, placement, of, right, new, ventriculoperitoneal, vp, shunt, stratum, valve, and, to, removal, of, right, frontal, ommaya, reservoir, indication, for, surgery, the, patient, is, a, month, old, infant, born, premature, with, intra...	title operation placement right new ventriculoperitoneal vp shunt stratum valve removal right frontal ommaya reservoir indication surgery patient month old infant born premature intraventricular hemorrhage ommaya reservoir recommendation removal repla...

2.0 Word Cloud

```
# Transcription for all
from wordcloud import WordCloud
def wordcloud(df):
    wordcloud_ent = WordCloud().generate(' '.join(df['lemmatised_without_stop']))
    plt.imshow(wordcloud_ent, interpolation='bilinear')
    plt.axis("off")
    return plt.show()
wordcloud(df)
```

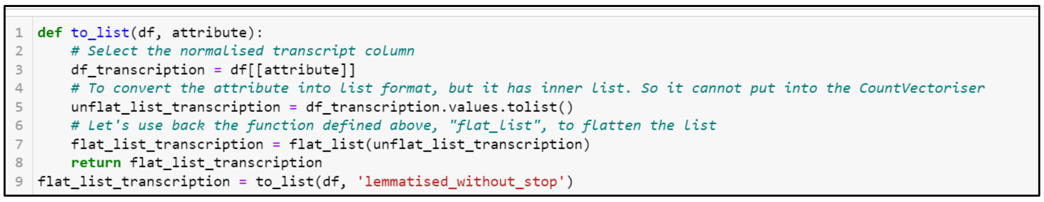


```
# Transcription for all
from wordcloud import WordCloud
def wordcloud(df):
    wordcloud_ent = WordCloud().generate(' '.join(df['lemmatised_without_stop']))
    plt.imshow(wordcloud_ent, interpolation='bilinear')
    plt.axis("off")
    return plt.show()
wordcloud(df)
```

3.0 N-Gram Features Extraction

3.1 Transform column values to a list

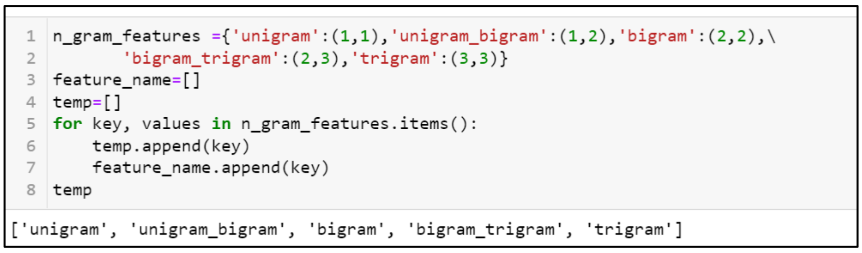
```
def to_list(df, attribute):
    # Select the normalised transcript column
    df_transcription = df[[attribute]]
    # To convert the attribute into list format, but it has inner list. So it cannot put into the CountVectoriser
    unflat_list_transcription = df_transcription.values.tolist()
    # Let's use back the function defined above, "flat_list", to flatten the list
    flat_list_transcription = flat_list(unflat_list_transcription)
    return flat_list_transcription
flat_list_transcription = to_list(df, 'lemmatised_without_stop')
```



```
def to_list(df, attribute):
    # Select the normalised transcript column
    df_transcription = df[[attribute]]
    # To convert the attribute into list format, but it has inner list. So it cannot put into the CountVectoriser
    unflat_list_transcription = df_transcription.values.tolist()
    # Let's use back the function defined above, "flat_list", to flatten the list
    flat_list_transcription = flat_list(unflat_list_transcription)
    return flat_list_transcription
flat_list_transcription = to_list(df, 'lemmatised_without_stop')
```

3.2 Set the n-gram range to be extracted

```
n_gram_features ={'unigram':(1,1),'unigram_bigram':(1,2),'bigram':(2,2),\
       'bigram_trigram':(2,3),'trigram':(3,3)}
feature_name=[]
temp=[]
for key, values in n_gram_features.items():
    temp.append(key)
    feature_name.append(key)
temp
```



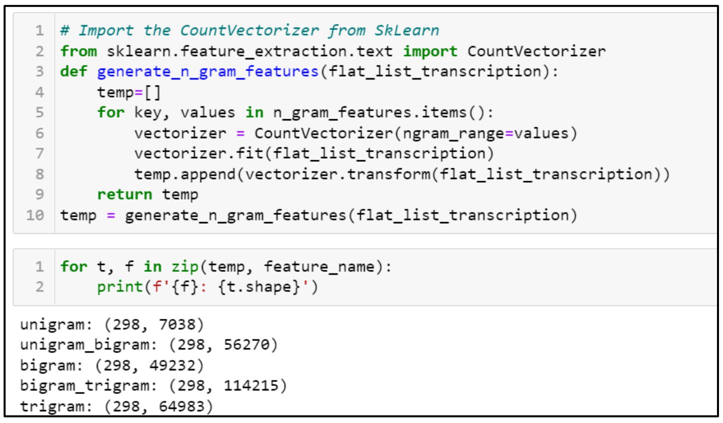
```
n_gram_features ={'unigram':(1,1),'unigram_bigram':(1,2),'bigram':(2,2),\
       'bigram_trigram':(2,3),'trigram':(3,3)}
feature_name=[]
temp=[]
for key, values in n_gram_features.items():
    temp.append(key)
    feature_name.append(key)
temp
```

```
['unigram', 'unigram_bigram', 'bigram', 'bigram_trigram', 'trigram']
```

3.3 Extract the 5 different n-gram features and stored it in a list

```
# Import the CountVectorizer from SkLearn
from sklearn.feature_extraction.text import CountVectorizer
def generate_n_gram_features(flat_list_transcription):
    temp=[]
    for key, values in n_gram_features.items():
        vectorizer = CountVectorizer(ngram_range=values)
        vectorizer.fit(flat_list_transcription)
        temp.append(vectorizer.transform(flat_list_transcription))
    return temp
temp = generate_n_gram_features(flat_list_transcription)

# show each n-gram features shape
for t, f in zip(temp, feature_name):
    print(f'{f}: {t.shape}')
```



```
# Import the CountVectorizer from SkLearn
from sklearn.feature_extraction.text import CountVectorizer
def generate_n_gram_features(flat_list_transcription):
    temp=[]
    for key, values in n_gram_features.items():
        vectorizer = CountVectorizer(ngram_range=values)
        vectorizer.fit(flat_list_transcription)
        temp.append(vectorizer.transform(flat_list_transcription))
    return temp
temp = generate_n_gram_features(flat_list_transcription)
```

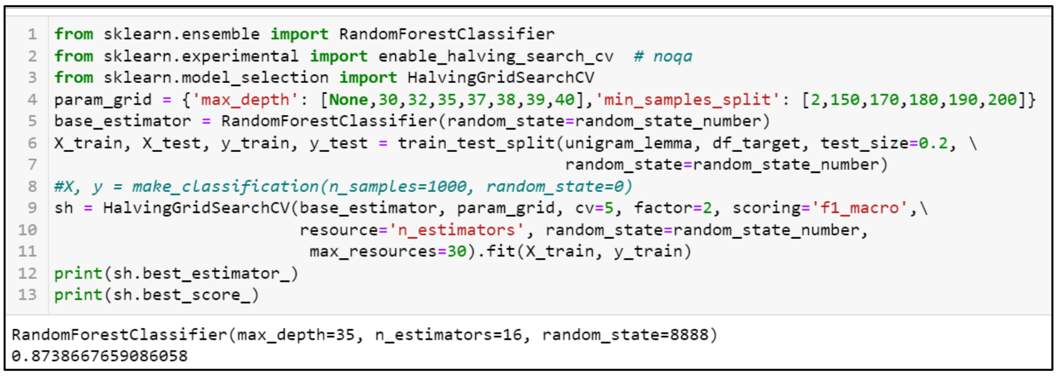
```
for t, f in zip(temp, feature_name):
    print(f'{f}: {t.shape}')
```

```
unigram: (298, 7038)
unigram_bigram: (298, 56270)
bigram: (298, 49232)
bigram_trigram: (298, 114215)
trigram: (298, 64983)
```

4.0 Tuning Hyperparameters of Random Forest Classifier using HalvingGridSearchCV

The search strategy starts evaluating all the candidates with a small number of resources and iteratively selects the best candidates, using more and more resources. The method allow to select parameters in much less time.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.experimental import enable_halving_search_cv  # noqa
from sklearn.model_selection import HalvingGridSearchCV
param_grid = {'max_depth': [None,30,32,35,37,38,39,40],'min_samples_split': [2,150,170,180,190,200]}
base_estimator = RandomForestClassifier(random_state=random_state_number)
X_train, X_test, y_train, y_test = train_test_split(unigram_lemma, df_target, test_size=0.2,
random_state=random_state_number)
#X, y = make_classification(n_samples=1000, random_state=0)
sh = HalvingGridSearchCV(base_estimator, param_grid, cv=5, factor=2, scoring='f1_macro',resource='n_estimators',
random_state=random_state_number,
                          max_resources=30).fit(X_train, y_train)
print(sh.best_estimator_)
print(sh.best_score_)
```



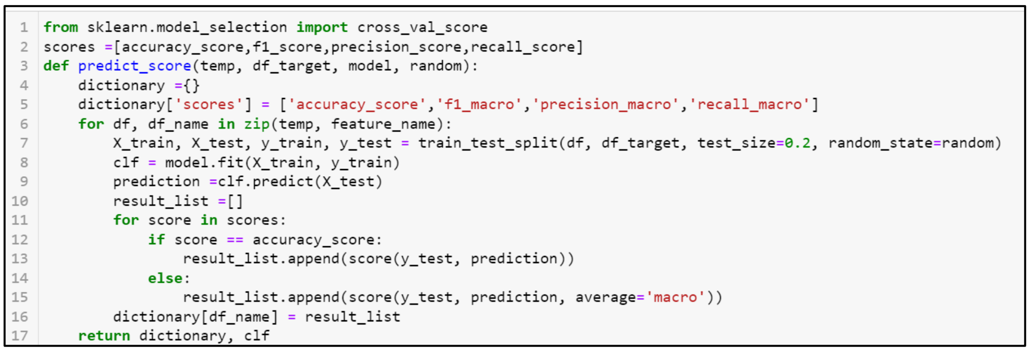
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.experimental import enable_halving_search_cv  # noqa
from sklearn.model_selection import HalvingGridSearchCV
param_grid = {'max_depth': [None,30,32,35,37,38,39,40],'min_samples_split': [2,150,170,180,190,200]}
base_estimator = RandomForestClassifier(random_state=random_state_number)
X_train, X_test, y_train, y_test = train_test_split(unigram_lemma, df_target, test_size=0.2, \
                                                    random_state=random_state_number)
#X, y = make_classification(n_samples=1000, random_state=0)
sh = HalvingGridSearchCV(base_estimator, param_grid, cv=5, factor=2, scoring='f1_macro',\
                         resource='n_estimators', random_state=random_state_number,
                          max_resources=30).fit(X_train, y_train)
print(sh.best_estimator_)
print(sh.best_score_)
```

```
RandomForestClassifier(max_depth=35, n_estimators=16, random_state=8888)
0.8738667659086058
```

5.0 Test Set Prediction

5.1 Define the prediction function

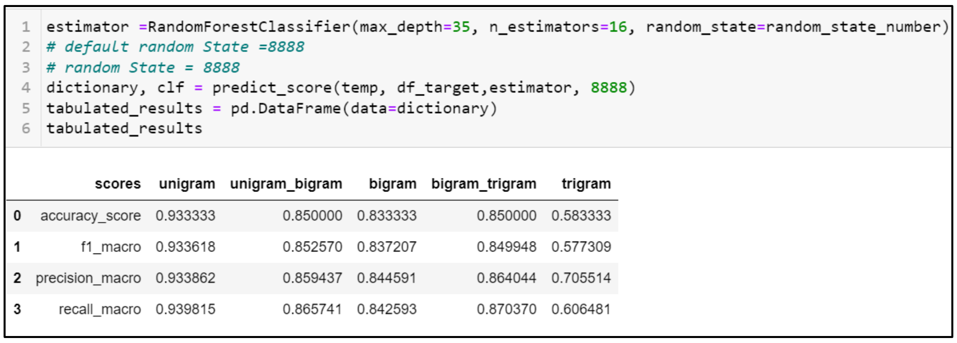
```
from sklearn.model_selection import cross_val_score
scores =[accuracy_score,f1_score,precision_score,recall_score]
def predict_score(temp, df_target, model, random):
    dictionary ={}
    dictionary['scores'] = ['accuracy_score','f1_macro','precision_macro','recall_macro']
    for df, df_name in zip(temp, feature_name):
        X_train, X_test, y_train, y_test = train_test_split(df, df_target, test_size=0.2, random_state=random)
        clf = model.fit(X_train, y_train)
        prediction =clf.predict(X_test)
        result_list =[]
        for score in scores:
            if score == accuracy_score:
                result_list.append(score(y_test, prediction))
            else:
                result_list.append(score(y_test, prediction, average='macro'))
        dictionary[df_name] = result_list
    return dictionary, clf
```



```
from sklearn.model_selection import cross_val_score
scores =[accuracy_score,f1_score,precision_score,recall_score]
def predict_score(temp, df_target, model, random):
    dictionary ={}
    dictionary['scores'] = ['accuracy_score','f1_macro','precision_macro','recall_macro']
    for df, df_name in zip(temp, feature_name):
        X_train, X_test, y_train, y_test = train_test_split(df, df_target, test_size=0.2, random_state=random)
        clf = model.fit(X_train, y_train)
        prediction =clf.predict(X_test)
        result_list =[]
        for score in scores:
            if score == accuracy_score:
                result_list.append(score(y_test, prediction))
            else:
                result_list.append(score(y_test, prediction, average='macro'))
        dictionary[df_name] = result_list
    return dictionary, clf
```

5.2 Make prediction and store the metric scores in a table

```
estimator =RandomForestClassifier(max_depth=35, n_estimators=16, random_state=random_state_number)
# default random State =8888
# random State = 8888
dictionary, clf = predict_score(temp, df_target,estimator, 8888)
tabulated_results = pd.DataFrame(data=dictionary)
tabulated_results
```

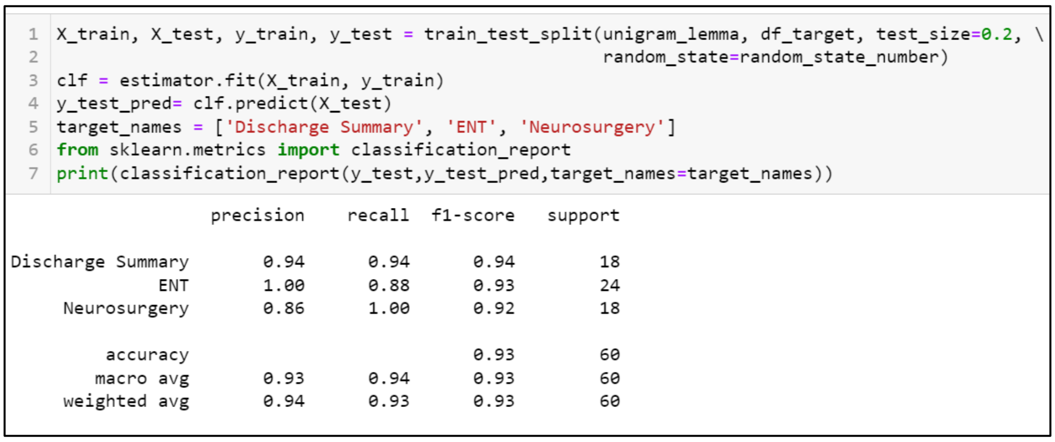


```python
estimator =RandomForestClassifier(max_depth=35, n_estimators=16, random_state=random_state_number)
# default random State =8888
# random State = 8888
dictionary, clf = predict_score(temp, df_target,estimator, 8888)
tabulated_results = pd.DataFrame(data=dictionary)
tabulated_results
```

	scores	unigram	unigram_bigram	bigram	bigram_trigram	trigram
0	accuracy_score	0.933333	0.850000	0.833333	0.850000	0.583333
1	f1_macro	0.933618	0.852570	0.837207	0.849948	0.577309
2	precision_macro	0.933862	0.859437	0.844591	0.864044	0.705514
3	recall_macro	0.939815	0.865741	0.842593	0.870370	0.606481

5.3 Make prediction on each target class

```
X_train, X_test, y_train, y_test = train_test_split(unigram_lemma, df_target, test_size=0.2,
random_state=random_state_number)
clf = estimator.fit(X_train, y_train)
y_test_pred= clf.predict(X_test)
target_names = ['Discharge Summary', 'ENT', 'Neurosurgery']
from sklearn.metrics import classification_report
print(classification_report(y_test,y_test_pred,target_names=target_names))
```



```python
X_train, X_test, y_train, y_test = train_test_split(unigram_lemma, df_target, test_size=0.2, \
                                                     random_state=random_state_number)
clf = estimator.fit(X_train, y_train)
y_test_pred= clf.predict(X_test)
target_names = ['Discharge Summary', 'ENT', 'Neurosurgery']
from sklearn.metrics import classification_report
print(classification_report(y_test,y_test_pred,target_names=target_names))
```

```
                   precision    recall  f1-score   support

Discharge Summary       0.94      0.94      0.94        18
              ENT       1.00      0.88      0.93        24
     Neurosurgery       0.86      1.00      0.92        18

         accuracy                           0.93        60
        macro avg       0.93      0.94      0.93        60
     weighted avg       0.94      0.93      0.93        60
```