University of Cape Town

Department of Statistical Sciences

Faculty of Science

Masters Half Dissertation

Performance Analysis of Text Classification Algorithms for PubMed Articles

Author:
Dr Suzana Savvi

Supervisor:
Mr Koen Bonenkamp
Prof Fransesca Little

SVVSUZ001
October 27, 2021

'I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own'

Signed by candidate

Contents

1 Abstract

The Medical Subject Headings (MeSH) thesaurus is a controlled vocabulary developed by the US National Library of Medicine (NLM) for indexing articles in Pubmed Central (PMC) archive. The annotation process is a complex and time-consuming task relying on subjective manual assignment of MeSH concepts. Automating such tasks with machine learning may provide a more efficient way of organizing biomedical literature in a less ambiguous way. This research provides a case study which compares the performance of several different machine learning algorithms (Topic Modelling, Random Forest, Logistic Regression, Support Vector Classifiers, Multinomial Naive Bayes, Convolutional Neural Network and Long Short-Term Memory (LSTM)) in reproducing manually assigned MeSH annotations. Records for this study were retrieved from Pubmed using the E-utilities API to the Entrez system of databases at NCBI (National Centre for Biotechnology Information). The MeSH vocabulary is organised in a hierarchical structure and article abstracts labelled with a single MeSH term from the top second two layers were selected for training the machine learning models. Various strategies for text multi-class classification were considered. One was a Chi-square test for feature selection which identified words relevant to each MeSH label. The second approach used Named Entity Recognition(NER) to extract entities from the unstructured text and another approach relied on word embeddings able to capture latent knowledge from literature. At the start of the study text was tokenised using the Term Frequency Inverse Document Frequency (Tf-idf) technique and topic modelling performed with the objective to ascertain the correlation between assigned topics (unsupervised learning task) and MeSH terms in PubMed. Findings revealed the degree of coupling was low although significant. Of all of the classifier models trained, logistic regression on Tf-idf vectorised entities achieved highest accuracy. Performance varied across the different MeSH categories. In conclusion automated curation of articles by abstract may be possible for those target classes classified reliably and reproducibly.

2 Introduction

Pubmed is a service provided by the United States National Library Medicine (NLM), under the U.S. National Institutes of Health (NIH) and contains over 23 million citations from Medline, a bibliographic database of biomedical literature. The Medical Subject Headings (MeSH) thesaurus is a hierarchy of terms maintained by NLM. MeSH is the main resource used by PUBMED to provide headings (terms) which are used to index the biomedical scientific bibliography in MEDLINE. (publicly available at https://www.nlm.nih.gov/mesh/meshhome.html) At present the MeSH vocabulary contains over 29 000 terms organised by tree hierarchy covering sixteen separate branches that can reach up to twelve levels of depth which allows searching medline citations at various levels of specificity [2] .

A rapid increase in the number of publications is evident in the last decade. Manually absorbing, interpreting, and curating the rapidly growing volumes of texts is going beyond the scope of individual persons. Currently, indexing is done manually and requires an in-depth knowledge of the extensive MeSH thesaurus [47]. It is a time consuming and laborious method relying on a comprehensive scope of each articles contents in order to identify the most relevant set of terms among the MeSH vocabulary. It is reported that on average, it takes 2 to 3 months for a new article to be indexed upon entering PubMed [18]. The issue can be addressed with computer algorithms, such as natural language processing (NLP) and subsequent text mining techniques, to process existing repositories of text abstracts from PubMed [3].

Automatic indexing by computers has been previously proposed and attempted but remains challenging, namely due to the large number and biased distribution of distinct main MeSH headings. In order to advance the state of the art in automatic MeSH indexing a community-wide shared task called BioASQ held a global challenge on large-scale biomedical semantic indexing in 2013 [31, 48].

2.1 Aims

This study aims to assess the performance of machine learning models for automatically indexing medical abstracts with a unique Mesh term from the second and third layer of the MeSH tree as illustrated in Fig.1. For this study we consider single labels for classification and therefore articles with more than one Mesh term were filtered out. Text classification differs from topic modelling in that it is supervised learning on labelled data. Instead of placing documents in predefined sets of classes, topic modelling is unsupervised learning, extracting cluster based classes from hidden topics in the corpus of documents. LDA (Latent Dirichlet Allocation) is a topic modelling algorithm that takes documents as input and finds topics as outputs [4]. Should there be any ambiguity in understanding the key topics from human decision making based on mesh_indexing, LDA may reconcile this by representing the article as a distribution over topics, and each topic as a distribution over words. In this study we compared this

Figure 1: Mesh Tree View (https://meshb.nlm.nih.gov/treeView)

topic modelling approach for cataloguing research articles based on abstracts , to mesh_indexed recommendations. Further we aimed to subsequently consider how supervised methods perform on the same corpus clustered by unsupervised topic modelling.
Benchmark classifiers considered for the multi-class text classification task included Logistic regression, Naïve-Bayes, Linear SVM and Random Forest. Word Vector based approaches applied for this natural language problem included Deep Learning models, namely Convolutional and Recurrent Neural Networks. Specifically, the latter sequence model used here is the commonly used variant LSTM.

For this study, the focus will be on text classification of research articles using only the available abstract content readily accessible on Pubmed. The study benefited from the manual assignments of MeSH terms to citations as training data with preassigned labels for our machine learning models. Correctly classified scientific research publications will allow for improved and expedited indexing which could facilitate improved Pubmed article retrieval for researchers. The workflow is shown in Fig. 2 and will act as a guideline for this study.

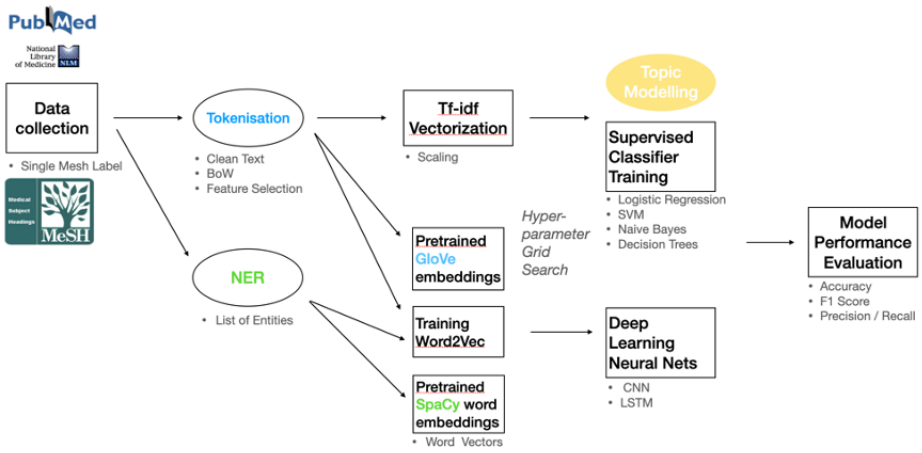

Figure 2: Methodology Flow Chart

2.2 Literature Review

Text preprocessing is a process that consists of a series of steps to clean, and standardize textual data into a form that can be readily used for machine learning and deep learning. Existing steps for normalising the text data include tokenisation (ie. splitting text into single terms) and several text cleaning techniques. Cleaning text is achieved through multiple actions by removing special characters and stopwords (removal of frequently occurring terms), stemming (ie. mapping related terms to a common base form) and pruning (removal of text with very high or very low frequencies, which might not be highly informative). These are methods of reducing dimensionality and improving classification accuracy by reducing noise of the data. The set of documents is known as the corpus. The set of all unique words used in any of the articles constitutes our bag-of-words, also called the vocabulary. In the bag-of-words model, a text document is represented by the set of words used in the document. Bag-of-words considers only the the occurrence of words in a document and disregards the order of the words. Each word is assigned a unique number which is used to encode a document as a fixed-length vector. The length is equivalent to the size of the vocabulary of known words, and contains each word's frequency count [40].

The data is represented as a matrix, in which each article is a row, each word is a column, and the entry in row i and column j contains the number of times word j appears in document i (many of these values will be zero, so we have a sparse matrix). A final column contains our response variable, indicating the type of article (mesh term entry). Frequency counts of the words used in a document can then be used to either cluster documents or as features in predictive modelling [12]. Replacing the term frequencies with new weighted values called 'term frequencies inverse document frequency'(Tf-idf) downscales irrelevant words that appear a lot across documents and upweights words likely to be useful.

Tf-idf is described for term (t_i) in document(d_j) in the equations:

$$Termfrequency(t_i) = Tf(t_i) = f_{t_i,d_j} \Big/ \sum_{t' \in d} f_{t',d} \tag{1}$$

$$InverseDocumentfrequency(t_i) = Idf(t_i) = log\frac{N}{n_t} = -log\frac{n_t}{N} \tag{2}$$

$$Tf - Idf(t_i) = Tf \times Idf \tag{3}$$

Equation one is the number of times that term t_i occurs in document d_j and normalised by the total number of words in the document, where f_{t_i,d_j} is the count of term t_i in document d_j.$t^{'}$ is the complement of term t.

Equation two describes the the inverse document frequency. It is the log normalised inverse fraction of the documents that contain the word. The total number of documents is N divided by the number of documents containing the term n_t. Therefore the idf of rare terms is high, and the idf of frequent terms low [41].

Equation three is the product of Tf and Idf .The Tf-idf is the weight score of term t_i that occurs in a document and is inversely proportional to the term frequency. In this study words were split into tokens (tokenisation) and text vectorised by the Tf-idf term weighted approach.

Bag-of words is where each word is represented by a large sparse vector and dependent on the size of vocabulary it is dealing with. An improvement over traditional ways of encoding such as this is word embeddings. An embedding representation of a word is by a low-dimensional dense vector which represents the projection of the word into a continuous vector space. The position of a word within the vector space is generated by analyzing large collections of text corpora and studying the co-occurrences of words in the documents. Words that have the same meaning have similar numerical vector representation. In this way related words that often occur together in documents and appear in a similar context would share similar coordinates in the vector space and appear in close proximity of one another [49].

2.3 Statistical Models

Classification models are supervised machine learning algorithms that are used to classify data points based on the categorises it has observed in the past. Each classifier is a supervised learning algorithm meaning it requires training data. Commonly used algorithms for text data classification include Logistic regression, Support vector machines (SVM) and Random forest. These techniques are broadly classified as discriminative rather than probabilistic [22]. Discriminative models model the decision boundary between the classes. Probabilistic classifiers such as Multinomial Naive Bayes predict a probability distribution over a set of classes rather than only outputting the most likely class of an input observation.

2.3.1 Naive Bayes

Naive Bayes classifiers have been successfully applied to NLP tasks and have been demonstrated to be fast, reliable and accurate in a number of applications of NLP [46]. They provide straightforward probabilistic prediction, are easily interpretable with few tuneable parameters. Naive Bayes algorithm is based on Bayes Theorem and assumes (naively) that the words within a document are unrelated given the class. This assumption of conditional independence between features makes Naive Bayes classifier highly scalable and suitable for very high-dimensional data. It manages the many dimensions by decoupling the conditional word distributions related to each class variable. This leads to each distribution being independently estimated as a single dimension distribution [40] . An example would be :

$$P(sentence from abstract) =$$
$$P(sentence|Mesh\_Class) \times P(from|Mesh\_Class) \times P(abstract|Mesh\_Class)$$

Below Naive Bayes models the joint distribution of the feature sentence X(x=sentence from abstract) and target Y(y=mesh_class), and then predicts the posterior probability given as $P(Y|X)$ as follows:

$$P(Mesh\_Class|sentence from abstract) =$$
$$\left(\frac{P(sentence from abstract|Mesh\_Class) \times P(Mesh\_Class)}{P(sentence from abstract)}\right)$$

Bayes rule uses probability to make predictions based on the *prior* i.e. P(Mesh_Class) knowledge of conditions that might be related, in order to produces *posterior* i.e. P(Mesh_Class— sentence from abstract) predicted probabilities [40].

Naive Bayes estimates the conditional distribution of each document belonging to a particular class, where the highest probability will be output. The multinomial Naive Bayes classifier is one Naive Bayes classifier variant used specifically for prediction and classification tasks where we have more than two classes and it is often a good choice as an initial baseline classification [46]. However as Naive Bayes assumes all the features to be conditionally independent, if some are not as is often the case with a large feature space, the prediction might be poor.

2.3.2 Logistic Regression

Logistic regression splits the feature space linearly, and typically works reasonably well even when some of the variables are correlated. It is a linear classification method that learns the probability of a sample belonging to a certain class using the logistic function (also called the sigmoid) to estimate the parameter values. These are the coefficients of all the features such that the overall loss is minimized when predicting the mesh_term categories in this case.

$$(Mesh\_Term = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n)$$

Such that$\{x_1, x_2, \ldots, x_n\}$ are our word features and we are trying to estimate the coefficients $\{\beta_1, \beta_2, \ldots, \beta_n\}$.

To obtain the class probability values with the logistic regression model we use a sigmoid function derived by taking the inverse of the log-odds or logit (*log*istic un*it*) function where $p(X)/(1-p(X))$ is called odds:

$$log - odds = log\frac{p(X)}{1-p(X)} = \beta_0 + \beta_1 X \tag{4}$$

$$p(X) = \left(\frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \right) \tag{5}$$

The probability $p(X)$ ranges from $0 < p < 1$ in the S-shaped curve [11].

For estimating the regression coefficients, instead of reducing errors (ordinary least squares) the focus is more on optimizing the coefficients for each feature using Maximum-Likelihood Estimation (MLE) [11]. MLE helps in maximizing the likelihood function. The likelihood function for binary outcomes is :

$$\mathcal{L}(\beta) = \prod_{i=1}^{N} (\frac{\pi(X = x_i)}{1 - \pi(X = x_i)})^{y_i} (1 - \pi(X = x_i)) \tag{6}$$

Where the probability of outcome $\pi(X = x_i) = p(X) =$ sigmoid function (equation 5). The regression coefficients that maximise the likelihood are determined by differentiation of the log-likelihood using optimization techniques such as gradient descent. Alternatively, we can minimise the cross-entropy error function:

$$CE(\beta) = -log(\mathcal{L}(\beta)) \tag{7}$$

Where logistic regression is used for multi-class classification, the MLE minimises the cross-entropy loss function for multinomial distribution [40].

2.3.3 Support Vector Machine

SVM find the maximal-margin-hyperplane (also known as optimal separating hyperplane) that divides the classes. This is the hyperplane that is furthest from the training observations. The points lying on the boundaries are called support vectors and the middle of the margin is the optimal separating hyperplane. In support vector machines cost(C) is a hyperparameter determining the penalty for misclassifying an observation when defining points on the wrong side of the margin by an amount of ξ_i^* :

$$\sum_{i=1}^{n} \xi_i^* \leq C \tag{8}$$

To avoid overfitting, set the cost setting higher, as decreasing this tuning parameter to have fewer violations given the narrowing margin would result in a classifier with high variance [11] and poor generalisation.

SVM classifiers reportedly perform well on many NLP tasks as they tend to have better generalisation capability on unseen data. Applications typically involve high dimensional although sparse feature vectors. This property allows SVM to easily learn a classification hyperplane in the feature space where the maximum width of the margin is optimised as a result of the training observations being widely distributed throughout. Another advantage of SVM for NLP classification tasks is that it is able to determine which features are

most useful by assigning different weights to each. A redundant feature would occur almost equally in all training examples and have a learned weight that would be irrelevant to the classifier[23].The stochastic gradient descent is often used for minimizing the loss function in SVM algorithms.

For a multi-class classification problem one vs all approach is followed, where each each binary classifier learns by separating each n classes from the other *n-1* classes. When predicting a class for a given observation the distance to each hyperplane is scored for each classifier and the maximum score is chosen for selecting the class label [40].

2.3.4 Random Forest

Decision Trees are built by recursive binary splitting of a feature space into regions in such a way to reduce the variability within the resultant branch. All the features j and all possible cut-points s are considered and the feature and s-value which minimise the variance most are selected for splitting into 2 regions (R1, R2)[11]:

$$R_1(j, s) = X|X_j < s \tag{9}$$

$$R_2(j, s) = X|X_j \geq s \tag{10}$$

The total variance or node purity can be measured by the classification error rate although more typically a Gini index or entropy is used: [11]:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{11}$$

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{12}$$

Equations 11 and 12 show the Gini index (G) and entropy (D) where $\hat{p}_{mk}$ is the proportion of observations in the mth region from the kth class [11]. The predominant observations from a single class in the terminal node represent the classification label.

A major drawback of decision trees is that since they are non-parametric, the more data there is, the greater the depth of the tree [40]. Such really huge and deep trees are prone to overfitting. Random Forest is an extension of decision trees.A random forest can reduce the high variance of such flexible models by combining the predictions of many decision trees into a single ensemble model, whereby the final/leaf node will be the majority class in classification problems [30]. For Random Forests only a random set of features for each split is chosen to reduce the correlation between individual classifiers. The trees are grown recursively in this way by choosing only a number of bootstrap samples from the training set [6].

2.4 Deep Learning Models

A deep learning neural network goes though an iterative training process where it learns to map a set of inputs to a set of outputs from training data. With neural networks, the objective is to minimise the cost or loss function using an optimisation algorithm such as gradient descent where weights are updated using the backpropagation error algorithm.

The cost function $J(w)$ contains a regularization term in addition to loss function and is a generalisation of the one used for binary logistic regression (Logarithm of the referred likelihood function in Equation 6). For multi-classification tasks the loss function used is Categorical Cross Entropy (CE):

$$J(w) = -\sum_{j=1}^{M}\sum_{i=1}^{N} y(x) log\, p(x) + regularisation\, term \tag{13}$$

where y is the true label and p(x) is the predicted label and M is the number of classes [32]. The estimated probabilty p(x) is calculated using the sigmoid function described in Equation 5. The regularisation term serves to minimise the magnitude of the weights. The objective is to minimise the loss using the cost function by doing backward pass to adjust the weights until convergence. Updating the weights this way uses the backpropagation of error algorithm:

$$\omega^{new} = \omega^{old} - learning\, rate \times \frac{\partial C}{\partial \omega} \tag{14}$$

where the derivative of the cost function is used for doing the gradient descent to calculate the new weight(ω) [17].Typically, stochastic gradient descent (SGD) is used to train a neural network.

Deep learning methods are achieving state of the art results for text classification [16]. Typically such results are achieved with the use of word embedding for representing words. Pretrained word embeddings that were trained on very large text corpora, offer good universal features for use in natural language processing [20]. The models leverage massive datasets and are built using a vast corpus of language that captures word meanings in a statistically robust manner. Example training data sets include word vectors induced from PMC full-text archive of biomedical and life sciences journal literature, or Wikipedia.

SciSpacy is an NLP library for Python with word embedding models built in [34]. It has a number of different pretrained models trained on the large corpus of biomedical and general-domain texts from PMC. For this study, the larger-than-standard en_core_sci_lg library was used, which includes 600k word vectors with 300 dimensions [28, 43].

Gensim is a topic modelling library for Python that provides pre-trained word embeddings, as well as allowing access to Word2Vec and other word embedding algorithms for training. GloVe (global vectors for word representation) is an unsupervised learning algorithm developed by Stanford for obtaining vector representations for words. 'Glove-wiki-gigaword-300' is a GloVe word vector model trained on an English Wikipedia dump and English Gigaword 5th Edition

dataset. For this study this more generalised model's word embeddings were loaded into Gensim library for comparative purposes. Its dimensionality is 100 and has 6B tokens (6 Billion tokens and 100 features uncased).

The disadvantage of pre-trained word embeddings is that the words contained within may not capture the domain-specific language. Training word embeddings from scratch for specific use-cases may be more relevant as results may not be optimal with pretrained emdeddings on account of being too general [43]. For this study word embeddings were randomly initialized and trained using our dataset. For this purpose the Gensim library provides a simple API to the Google word2vec algorithm to train ones own word vector models.

2.4.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is a deep learning architecture, and although originally built for image processing, CNN has also been effectively used for text classification as they easily integrate with pretrained word embeddings [26, 22]. The non-linearity of the network and the ability to pick out salient features to detect position-invariant patterns lead to superior classification accuracy. Yoav Goldberg describes CNN aptly by saying *The CNN is in essence a feature-extracting architecture. The CNNs layer's responsibility is to extract meaningful sub-structures that are useful for the overall prediction task at hand.* [15, 14].

Briefly, CNN sentences are mapped to embedding vectors and are available as a matrix input to the model. Convolutions are performed across the input word-wise with a set of kernels of size $d \times d$.

The resulting feature maps are then processed using different pooling techniques to reduce outputs while preserving important features. The most common pooling method is max pooling where the maximum element in the pooling window is selected [5].

A convolutional architecture that alternates convolutional layers with pooling layers is used to generate a feature map over the sentence successfully capturing long-range dependencies between words [26]. Next the pooled output from the stacked featured maps are flattened into one column. The final fully connected layers of the CNN are where the extracted features are interpreted in terms of predictive output [35].

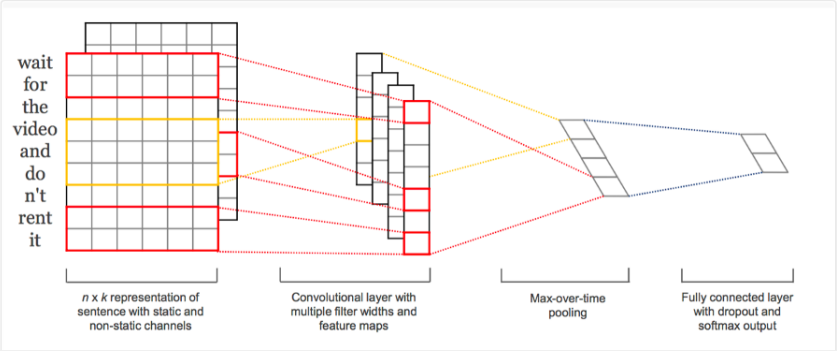

Figure 3: An example of CNN Filter and Pooling Architecture for NLP [20]

2.4.2 LSTM

Another neural network architecture that researchers have used for text mining and classification is Recurrent Neural Networks (RNN). RNN are designed to handle sequences of words with the intention to uncover text structure from information based on word dependencies. This makes RNN a powerful method for text, string, and sequential data classification [22]. One commonly used variant of sequence models, LSTM not only takes the new input (word) but also considers the output (words) of the last layer into consideration while modeling. LSTM is particularly useful with respect to overcoming the vanishing gradient problem and preserving long term dependencies. This allows for better semantic analysis of a data set's structure.

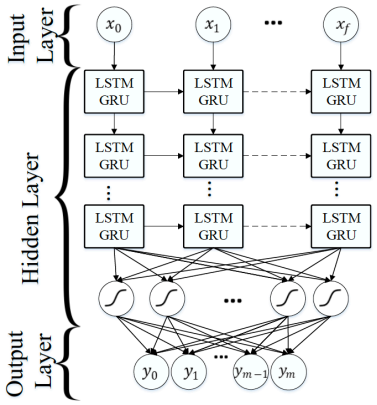

Figure 4: LSTM schematic

LSTM uses multiple gates known as Gated Recurring Units (GRU) (see Fig.4) to remove or add information to each node state. A bi-directional LSTM propagates the input forward and backwards through the LSTM layer and then concatenates the output. This helps the LSTM to learn long range dependencies. The final prediction is made by combining the results of both the BiDirectional layer and final fully connected dense layer of the network [9].

2.5 Objectives

The objective for this study is to evaluate multi-class classification models for scientific text documents, using 2 main strategies:

- Machine learning with bag of words (Tf-idf text vectorisation) feature engineered text;
- Deep learning with word embedding (Word2Vec)

As prediction accuracy is largely influenced by the type of data used [54], both discrimination and probabilistic learning methods were considered. This study aims to explore a range of word usage approaches and several statistical models, to examine which analysis may be best suited for classification of Pubmed articles. The models selected for this study include SVM, logistic regression, decision tree forests and Naive Bayes, and two deep learning models, CNN and LSTM. These algorithms were selected for text classification on account of achieving a good prediction accuracy in previous literature reviews. Further model performance is optimised to improve accuracy by fine tuning hyperparameters. NLP libraries were used.

3 Exploring and Preprocessing the Data

Data sets were downloaded using E-utilities, the public API to the NCBI Entrez system which allows access to all Entrez databases including PubMed. The E-utilities are a suite of eight server-side programs that accept a fixed URL syntax for search, link and retrieval operations. For the purposes of this study the root node of the Mesh_tree is level 0, and the second and third layers of the MeSH tree are level 1 and level 2 respectively.

- Level 1: The dataset downloaded contains 14661 research abstracts, and classified into 17 categories (Fig. 5 left barplot).
- Level 2: The dataset downloaded contains 11082 research abstracts, and classified into 15 categories(Fig. 5 right barplot).

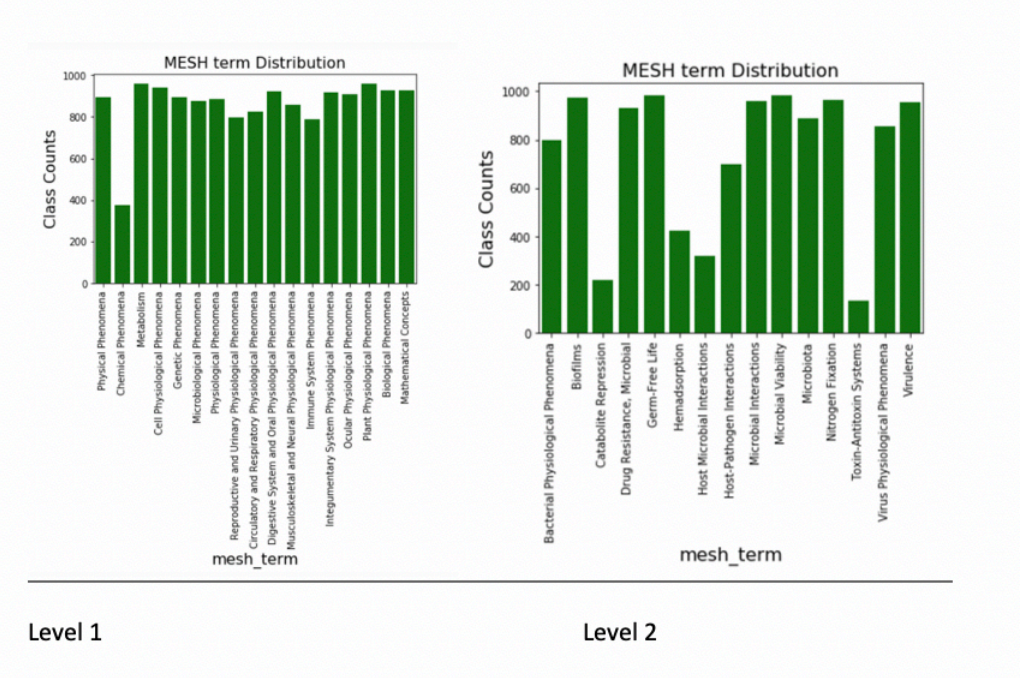

Figure 5: Data Distribution

High volume datasets such as used in this study contain large word vocabularies which translates to a the high number of dimensions along which the data is distributed. For this purpose two methods of data dimensionality reduction, which allow easier visualization of complex data were used. The first commonly used technique called t-distributed stochastic neighbor embedding

(t-SNE) is useful for exploring local neighborhoods and finding clusters in non-linear datasets. Here the focus is more on representing local similarities between individual points while with the second popular method, Principal Component Analysis (PCA) the focus is instead on preserving the global data structure.

3.1 Principal Component Analysis

PCA is an unsupervised dimensionality reduction technique which reduces high-dimensional interrelated data to a new set of variables called Principal Components. The final Principal Components are a linear combination of the original features where most of the information within the initial variables is squeezed or compressed into the first components.

To capture as much information from the original dataset PCA uses the correlated variables and tries to provide a smaller number of uncorrelated variables that keep the maximum amount of variation. The first component has the largest variance followed by the second component and so on.

This can be done multiple ways. Either by Singular value decomposition of the data matrix or Eigen decomposition of the covariance matrix. The covariance matrix is not more than a table that summarises the correlations between all the possible pairs of variables, and in the case of z-scored data the covariance is equal to the correlation matrix, since the standard deviation of all features is 1.

The first step of Eigen decomposition is to compute a covariance matrix which is simply a table that summarises the correlations between all the possible pairs of variables. Next the eigenvectors (unit vectors also known as the principal component weights or loadings) and their associated eigenvalues (scalars by which the eigenvector is multiplied) of the covariance matrix are calculated. The top N eigenvectors (based on their eigenvalues) are selected to become the N principal components. Eigen decomposition uses matrix multiplication (dot product) of normalized data by eigenvectors of the covariance matrix to project (transform) the data onto the reduced PCA space[42]. For the purpose of dimensionality reduction, we can project the data points onto the first n principal components as the representation of the data:

$$X_k = XW_k \tag{15}$$

where each principal component is the dot product of its weights (W=eigenvector) and the mean centered data(X) [37].
The importance of each feature corresponds to the absolute values in the eigenvectors. The larger the value the more a specific feature contributes to that principal component. Selecting features in this way according to their coefficient loadings makes PCA a particularly useful tool for feature selection where multicolinearity exists. If the variables are not correlated, PCA will be unable to determine principal components. PCA is used when there is a reduced number of features representing the maximum variance of the data. By taking all the features and reducing them down to just a few important principal ones which retain most of the variation, it is easy to project them onto a 2 or 3 -dimensional

space to visualize. In a biplot, a two-variable scatterplot showing information on both samples and variables, the length of the principal components reflects the amount of variance contributed.

Alternatively, these reduced components can be used in subsequent machine learning pipelines. The cumulative variation cutoff to retain principal components for analysis is 70% [19]. For NLP this variance is often spread over a wide number of principal components, each commonly accounting for a small fraction of the cumulative variance. This is because words or tokens from text have small discriminative power individually (see Fig.14(iii) and Fig.15(iii)). For this reason it is preferred in NLP to instead preprocess text linguistically and not reduce dimensions mathematically. Dropping the stop-words, stemming or lemmatising the rest of the words, and dropping words which occur less than n times will reduce the dimensionality without loosing information.

3.2 t-Distributed Stochastic Neighbor Embedding

PCA assumes a linear relationship between features. The algorithm is not well suited to capturing non-linear relationships. In this respect t-SNE, a probabilistic nonlinear dimensionality reduction technique provides better results than PCA at modelling curved manifolds and interpreting complex polynomial relationship between features. This makes it well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions.

While global approaches such as PCA focus on preserving geometry by keeping the distances between widely separated data points, t-SNE preserves the clustering of similar components which serves to retain information while reducing the dimensions. It is uniquely capable of retaining both the local and global structure of the data at the same time.

Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities. This way the algorithm constructs a probability distribution for the entire dataset in both a high and low dimensional space and tries to minimize the sum of the difference of the probabilities. It does so by applying a gradient descent method to the Kullback-Leibler divergence function, until the distributions are as close as possible [17]. The 't' in the name refers to the student-t distribution that is used to overcome the crowding issue when points are mapped to the lower dimension. The thicker tail means that when comparing probabilities the distance between points is larger which preserves the local structure [50, 17].

The best way to use the algorithm is to use it for exploratory data analysis as it gives a very good sense of patterns hidden inside the data.

Word and document vector representations (refer to 3.3 section next) capture many linguistic and semantic properties for text comparison. To gain a bird's-eye view of different texts, dimension reduction using t-SNE algorithm computes a 3D visual to be explored like a geographic map for similarities and differences in textual properties. By identifying observed clusters based on similarity of

data points with multiple features, patterns in the data are discovered. This technique is ideal for data exploration and visualization, although in mapping the multi-dimensional data to a lower dimensional space the input features are no longer identifiable.

3.3 Doc2Vec and K-Means Clustering

Each abstract was represented as a weighted combination of all words in that abstract. If each word of the text abstract is represented by an n dimensional vector, the abstract itself can be represented by a vector of this size by averaging element wise over the n dimensions [12]. We leveraged Spacy for averaged embeddings from all words in each document. The document similarity score was calculated using the document vectors. t-SNE was used to visualise these embeddings.

t-SNE models each high-dimensional document vector to its 2 or 3 dimensional point in such a way that similar objects have nearby points and dissimilar objects are distant. The original algorithm uses the Euclidean distance between objects as the base of its similarity metric (Fig.16).

The Doc2Vec averaged embeddings for words in each document were used as document features for our corpus by K-Means clustering to cluster our documents. The K-Means clustering algorithm initialises k number of centroids and tries to partition the data by allocating every data point to the nearest cluster. The algorithm tries to minimise the within-cluster sum-of-squares, a criterion otherwise known as inertia. This algorithm is considered one of the most popular clustering algorithms due to its ease of use and ability to scale big data environments [40].

3.4 Topic Modelling

Topic modelling is an unsupervised machine learning technique used in NLP for text mining a corpus of documents to extract key themes or concepts which are represented as abstract 'topics'. Each topic can be represented as a bag or collection of words/terms from the document corpus. There are various techniques for topic modeling and most of them involve some form of matrix decomposition. One such example is Latent Dirichlet Allocation (LDA) which is a generative probabilistic model in which each document has a 'mixture' of a fixed number of topics. Specifically Bayesian inference associates each document as a probability distribution over topics, and topics as a probability distributions over words. LDA uncovers the latent topics contained within the corpus based on the likelihood of word co-occurrence [4].

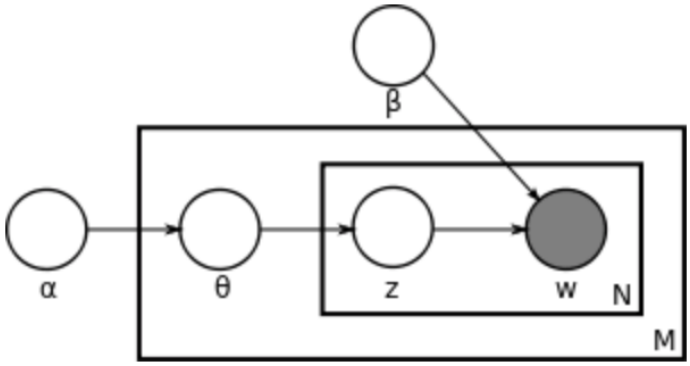

Figure 6: Plate Notation for representing the LDA model from *https* : *//en.wikipedia.org/wiki/Latent_Dirichlet_allocation.* The plates are repeated entities represented as boxes. The outer plate represents documents, while the inner plate represents the repeated word positions in a given document. The grey filled circle means that words are the only observable variables, the rest are latent.

The model parameters in Fig.6 are defined as follows [4]:

- M denotes the number of documents
- N is number of words in a given document
- α is the probability distribution of topics per document
- β the probability distribution of the number of words per topic
- θ is the topic distribution per document M
- z is the topic for the n-th word in document m,
- w is the specific word

To find the hidden topics in a corpus, documents are created by a generative process where the goal is to infer the words related to a given topic using the observed words in the corpus. The generative process of LDA follows a few steps to extract K topics from the documents [40] :

1. Initialize the hyperparameters α and β (these are the parameters of the prior probability distributions.
2. For each document, randomly initialize each word to one of the K topics.
3. Run through many iterations each time computing p(topic—document) and p(word—topic) and updating the prior probability.

After several iterations each document will be a mixture of topics. The LDA model was visualised using the package pyLDAvis which allows for comparing topics on two reduced dimensions and observing the distribution of words in topics.

Topic Coherence measures the semantic similarity between words within a topic. These measurements help distinguish whether topics are structured in meaningful ways. In order to evaluate the optimal number of topics we considered LDA models for different number of topics (k). Choosing the one that gives the highest coherence value usually offers meaningful and interpretable topics [38].

3.5 Comparing Topics and Clusters with Mesh Labels

The Mutual Information (MI) is a measure of the similarity between two labels of the same data, ignoring permutations. The MI score expresses the extent to which observed frequency of co-occurrence differs from what we would expect [39].

This metric was used in order to determine whether the match-up between the inferred topics and the assigned MESH labels was better than random chance. Statistically it measures the strength of association between the 2 cluster sets. Adjusted Mutual Information (AMI) is a variation of mutual information to normalise against chance. It takes into account that mutual information is generally higher for two clustering's with a larger number of clusters, regardless of whether there is actually more information shared. Specifically, AMI is used when the ground truth clustering is unbalanced and there exist small clusters, as is the case [52].

4 Classification

In order to see how useful the Bag of Words features are for classifying Pubmed abstracts, we trained a classifier based on them. Firstly the Bag of Words was used to build the feature matrix with each cell of the matrix having counts of words. This term frequency(tf) was then used to normalise the matrix. Once the content of each string has been converted into a vector of numbers using Tf-idf vectorizer, capturing both unigrams and bigrams, the data is able to be used for machine learning.

4.1 Feature Selection

A clean dataset will allow a model to learn meaningful features and not overfit on irrelevant noise. Less redundant data means less opportunity to make decisions based on noise,ultimately improving modeling accuracy. For this purpose, feature selection was carried out in one of 2 ways in order to drop some columns and reduce matrix dimensionality.

1. Feature selection was performed where only the most important features for each class were used. Here only the features with a certain p-value from the Chi-Square test were selected as the subset of relevant variables [13, 25]. The free software machine learning library Sklearn.feature_selection.chi2 was used to find the terms that are most correlated within each of the mesh classes. The number of features was significantly reduced by keeping the most statistically relevant ones.

2. Named Entity Recognition(NER) is a technique for extracting information that locates and classifies entities into predefined classes, also recognised as entity chunking/extraction. In scientific text mining for example, biomedical text terms such as protein or cell type would be assigned a named entity tag. For NER, ScispaCy, a full open-source SpaCy pipeline was used for identifying keyword entities. In this instance only extracted terms, with a valid concept in the ontology , are taken into account, disregarding position in the text or syntax.

The main NER model in the scispaCy package 'en_core_sci_lg' which was used in this study, is trained on the mention spans in the MedMentions dataset [33] which is is the largest annotated resource for the recognition of biomedical concepts [29]. The NER model as a result is able to recognize a wide variety of entity types along with non-standard syntactic phrases such as verbs and modifiers. The model rather than predicting specific entity types identifies more general spans which may include any entity in the biomedical database UMLS (Unified Medical Language System) [21] (see Fig.12).

4.2 Multi-Class Imbalanced Classification

Initial attempts were directed at resampling with a focus on balancing the skewed class distribution. Sampling with replacement was performed while fetching

abstracts from Pubmed. However as a significant number of the Pubmed_ids contained no abstracts, after filtering these out [dtf.abstract.isna()], the dataset remained unbalanced.

As an alternative approach, algorithms can be modified to change the way learning is performed to bias towards those classes that have fewer examples in the training dataset. This is called cost-sensitive learning for imbalanced classification where the cost matrix is defined using the class distribution. The scikit-learn library provides an optional class_weight argument modifying machine learning algorithms to make use of the cost matrix. In this study cost-sensitive learning was supported using this cost-sensitive extension to automatically calculate a class weighting and ensure each class has an equal weighting during the training of the model.

In addition the training, validation and/or test datasets were stratified, meaning that each fold of the cross-validation split had the same class distribution as the original dataset.

Another approach to addressing imbalanced datasets is to oversample the minority class. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique (SMOTE). SMOTE was used to oversample the minority class to balance the class distribution. Deep neural net algorithms were trained using the SMOTE-transformed training datasets in this study.

4.3 Classifier training

The supervised models selected for this study include SVM, Naive Bayes, logistic regression and Random Forests. These algorithms were selected as previous studies suggest they should achieve good prediction accuracy for text classification [22]. Five fold cross validation (out-of-sample testing) for training and evaluation was done as in Fig.7.

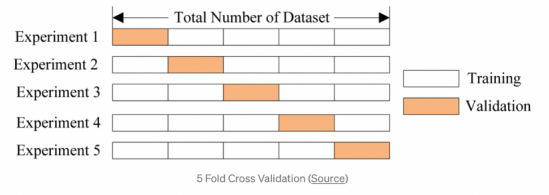

Figure 7: Cross Validation: training proceeds on the training set, after which evaluation is done on the validation set.

4.4 Deep Learning Models

For neural networks, the data was partitioned for learning the models into train/validation and test sets. A 0.7 -0.3 train-test split ratio was used to estimate the performance of deep learning algorithms and used to make predictions on data not used to train the model. Because the labels are imbalanced, we split

the data set in a stratified fashion. The features that were used to represent each article were unigrams and bigrams extracted from the abstracts of each article.

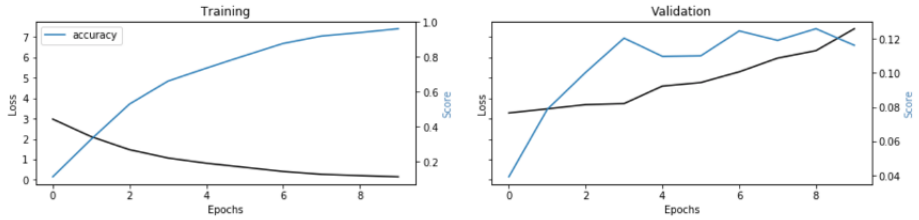

Figure 8: An example learning curve for CNN: The plots suggest that the model has a slight over fitting problem potentially caused by artificial balancing. Blue:Accuracy ; Black:Loss

For this multi-classification task the Sparse Categorical Cross Entropy (SCCE) cost function was used. This loss function does not need to one hot encode the target vector as classes are instead indexed.

The deep learning algorithms contain several hyperparameters to tune. A grid search of hyperparameters was used to explore the effects on model performance. The rule-of-thumb guide by researchers was used to determine ranges of the hyperparameters. (Refer to Appendix Table 29). Some parameter variations considered for neural network design included:

- Chosen Vocabulary. Cleaned text and extracted signature entities were used separately.
- Pretrained embeddings were compared to using custom embeddings trained on the gathered data.
- Number of layers, filters and kernel sizes.

For this study parameters were tuned for predictors to perform optimally.

An alternative to the SGD optimization algorithm was used. Adaptive Moment Estimation (ADAM) was selected for optimisation where the learning rates are adapted for each weight parameter. Smaller updates are performed for weight parameters associated with frequently occurring features (i.e. low learning rates) while larger updates are performed for infrequent features. This method improves the robustness of SGD, making it suitable for sparse large scale NLP datasets [1].

CNN Design:

The first layer of the CNN architecture is an embedding layer for input of the data set from our document word matrix. The height of the output matrix is the length of one document, set to 100 words. The width is 200 as each word in the document is encoded as a 200 element vector (embedding). As illustrated in Fig.

9, a conservative CNN configuration was used with the first 1D layer containing 32 filters (parallel fields for processing or interpreting the words) and a kernel size of 8. This means that the input will be processed 32 times resulting in 32 feature maps. The kernel size is the number of time steps the input sequence is read in or processed onto the feature maps.

The kernel size and input length of the matrix determine the weights for each filter. With the height of 100 and a kernel size of 8, the window will slide through the data 93 times ($100 \times (8 - 1) = 93$). Each column of the output matrix holds the weights of one single filter. Therefore 32 columns in total (width) by 93 (height). The result from the first CNN layer is passed through a rectified linear (relu) activation function and fed into the second CNN layer.

Using two convolutional layers in series allows the model to learn more complex features from the input data. Therefore another CNN layer was added with 32 different filters to be trained. Following the same logic as the first layer, the output matrix will be of size 86 x 32.

This is followed by a pooling layer which reduced the output of the convolutional layer by half. This serves to reduce the complexity of the output and prevent overfitting of the data. Subsequently another sequence of one dimensional (1D) CNN layers were added in order to learn higher level features. The output matrix after those two layers is a 39 x 64 matrix. One more pooling layer follows to avoid overfitting. The pooling layer reduces the learned features to half their size, consolidating them to only the most essential elements. Next, the learned features extracted by the CNN are flattened to one long vector followed by a dropout layer for regularization. As CNNs learn very quickly the dropout layer is intended to slow the learning process to improve the model. The output is passed through a fully connected (dense) layer before the final prediction layer. These final 2 layers are the back-end of the model similar to a standard Multilayer Perceptron which provides a buffer between the learned CNN features and the output for interpreting the learned features before making a prediction. In the final layer the output is reduced to a vector of length 15 or 17. This is equivalent to the corresponding number of mesh classes in each layer of the mesh-tree examined that we want to predict. The output layer uses a sigmoid activation to predict the probability of each mesh category.

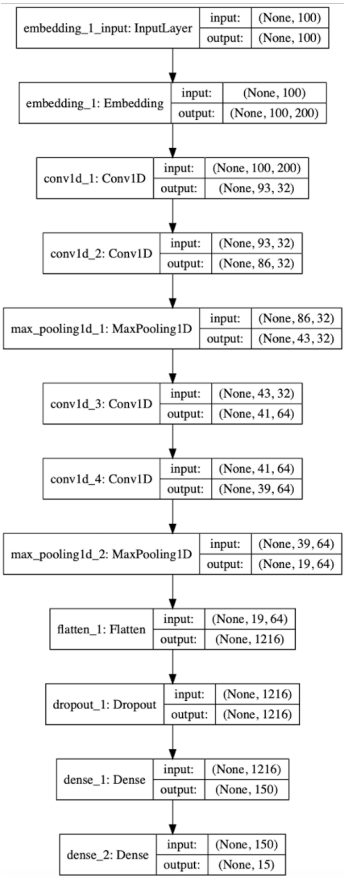

Figure 9: Plot of the defined CNN classification model.

LSTM: For this study the sequential model (Fig 10) was structured as follows:

- An Embedding layer that takes the text sequences as input and 200 (or 300) length word emdebbings as weights, as previously described for CNN.
- Two layers of Bidirectional LSTM, each with 100 memory units to model the order of words in a sequence in both directions.
- The final dense layer outputs 15 (or 17) probability values after softmax activation, one for each mesh class.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
embedding_1 (Embedding)      (None, 100, 300)          25412400
_________________________________________________________________
bidirectional_2 (Bidirection (None, 100, 200)          320800
_________________________________________________________________
bidirectional_3 (Bidirection (None, 200)               240800
_________________________________________________________________
dense_2 (Dense)              (None, 64)                12864
_________________________________________________________________
dense_3 (Dense)              (None, 17)                1105
=================================================================
Total params: 25,987,969
Trainable params: 575,569
Non-trainable params: 25,412,400
```

Figure 10: An example of a model summary for a LSTM model as extracted from Keras.

4.5 Model Performance Evaluation: Performance Metrics

To evaluate the performance of the training models the following accuracy metric was used. Classification accuracy is the fraction of predictions the model got right. The ratio between correctly predicted outcomes and the sum of all predictions made for a dataset

$$\frac{TruePositive(TP) + TrueNegative(TN)}{TP + TN + FalsePositive(FP) + FalseNegative(FN)}$$

This was displayed as :

- Confusion Matrix: Also known as an error matrix. It is a summary table for evaluating the performance of classification algorithms by breaking down the number of correct and incorrect predictions by each class.
- ROC: a plot that illustrates the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The area under the curve (AUC) indicates the probability that the classifier will correctly distinguish between classes. The ROC curve is plotted with TPR (y-axis) against the FPR (x-axis) (See Fig. 11).

Classification accuracy is not entirely appropriate for imbalanced classification problems. In this case the preference is to use precision and recall metrics for measuring performance [7].

- Precision quantifies the number of correct positive predictions made. All true positives divided by all positive predictions $TP/(TP + FP)$.

- Recall also known as sensitivity or TPR quantifies the number of correct positive predictions made out of all positive predictions that could have been made. True positives divided by all actual positives $TP/(TP+FN)$.

- F1-score: This is the weighted average of precision and recall (2*(recall x precision) / (recall + precision)).

For imbalanced classification problems, the minority class is often referred to as the positive outcome. In this way, precision and recall make it possible to assess the performance of a classifier on the minority class.

A precision-recall curve (or PR Curve) is a plot of the precision (y-axis) and the recall (x-axis) for different probability thresholds (See Fig. 11). The score is a value between 0.0 and 1.0, 1.0 for a perfect classifier and of 0.0 for no precision/recall. Notably, the PR curve focuses on the minority class, whereas the ROC curve covers both classes [7].

Generally you can say that the closer a PRC is to the upper right corner, the better the test is, and the closer a ROC curve is to the upper left corner, the better the test is.

Curve	x-axis		y-axis	
	Concept	Calculation	Concept	Calculation
Precision-Recall	Recall	TP / (TP + FN)	Precision	TP / (TP + FP)
ROC	1-specificty	FP / (FP + TN)	Sensitivity	TP / (TP + FN)

Figure 11: Precision Metrics

5 Results

To use data for machine learning, it is necessary to convert the content of each string into a vector of numbers with Tf-idf vectorizer. Each token is assigned a numerical id and then each sequence of 100 (word_ids) was made the same length by padding shorter sequences, and truncating longer sequences. The bag of words was used to build the feature matrix, using either 'clean_text' or 'entities' (see Fig.12). Fig.13 is an example of the working Dataframe. Each cell of the final matrix is counts of words. Tf-idf is used to normalize the matrix.

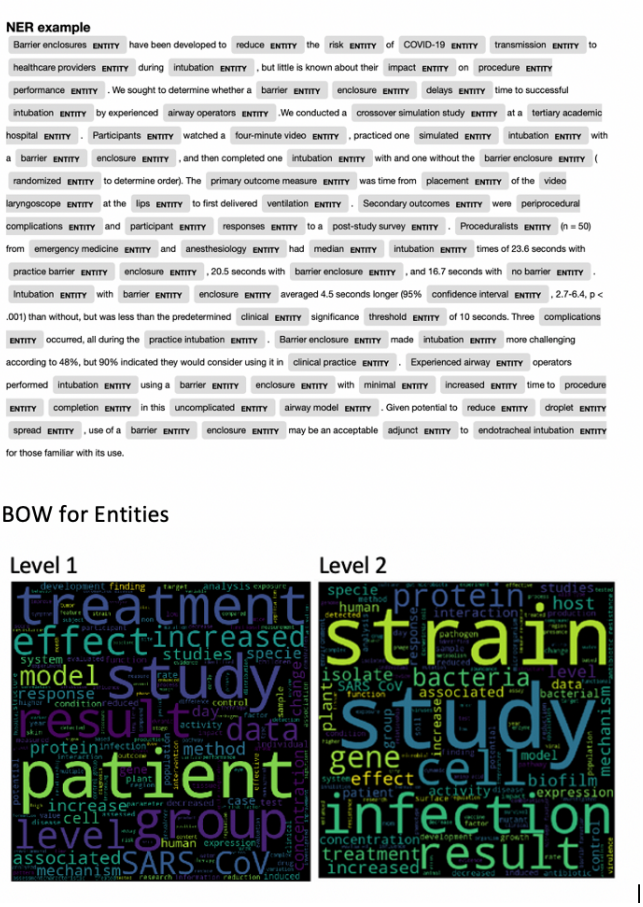

Figure 12: Extracted Entities

	pubmed_id	mesh_term	abstract	bow_length	dominant_topic	topic_probability	mesh_term_index	clean_text	entities
0	33034170	Bacterial Physiological Phenomena	The adhesion ability of microorganisms to the ...	108.0	3.0	0.343099	0	adhesion ability microorganism surface titaniu...	adhesion microorganisms surface titanium minip...
1	33033258	Bacterial Physiological Phenomena	Natural transformation is the process by which...	91.0	5.0	0.500676	0	natural transformation process bacteria take g...	Natural transformation process bacteria geneti...
2	33028835	Bacterial Physiological Phenomena	Type IVa pili are ubiquitous and versatile bac...	66.0	5.0	0.689453	0	type iva pili ubiquitous versatile bacterial c...	Type IVa pili ubiquitous bacterial cell filame...
3	32970667	Bacterial Physiological Phenomena	Mycetoma, a chronic infection of the skin and ...	111.0	9.0	0.351834	0	mycetoma chronic infection skin underlie struc...	Mycetoma chronic infection skin structures rel...
5	32936818	Bacterial Physiological Phenomena	Subclinical infection with Mycobacterium lepra...	108.0	4.0	0.339141	0	subclinical infection mycobacterium leprae one...	Subclinical infection Mycobacterium leprae lep...

Figure 13: DataFrame Example

5.1 Visualising High Dimensional Data

In order to reduce the number of dimensions a Chi-square test for association was performed on preprocessed 'clean' text. Chi2 features significantly associated ($p < 0.05$) with each mesh category were used for PCA.

- Level 1: 1 216 630 features reduced to 20 683. For PCA the cumulative explained variation for 1500 principal components was 81.2%. (Fig. 14(iii))
- Level 2: 895 587 features reduced to 22 475. For PCA the cumulative explained variation for 1500 principal components was 92.1%. (Fig.15(iii))

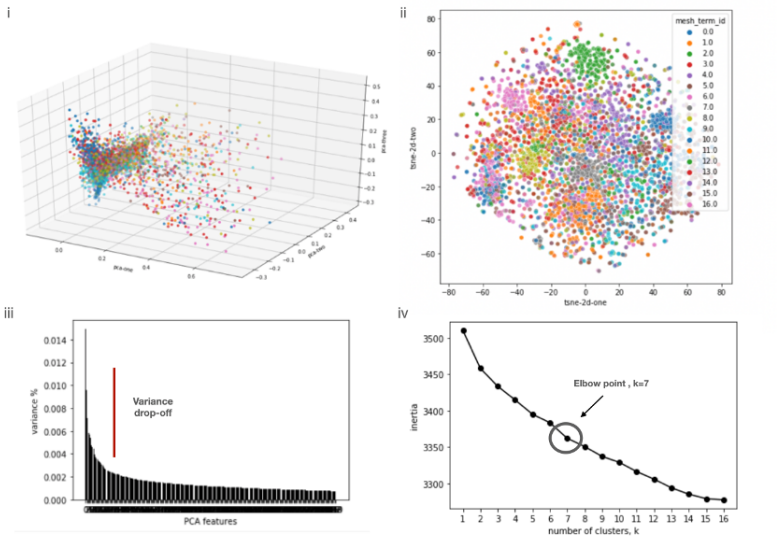

Figure 14: Dimension Reduction for Visualisation Level1

Fig.14 Level 1 of the Mesh Tree Hierarchy:

i PCA , first 3 principal components extracted and displayed on 3D scatter plot. Colours correspond to mesh classes.

ii T-SNE with PCA initialisation. Colours correspond to mesh classes.

iii Scree plot showing variance drop-off after the third component. Cumulative explained variation for 250 principal components: 0.385

iv K-means elbow plot to identify optimal cluster number using 1500 Principal components which explain 80% of the variance. The slight inflection point is circled denoting the 'elbow'.

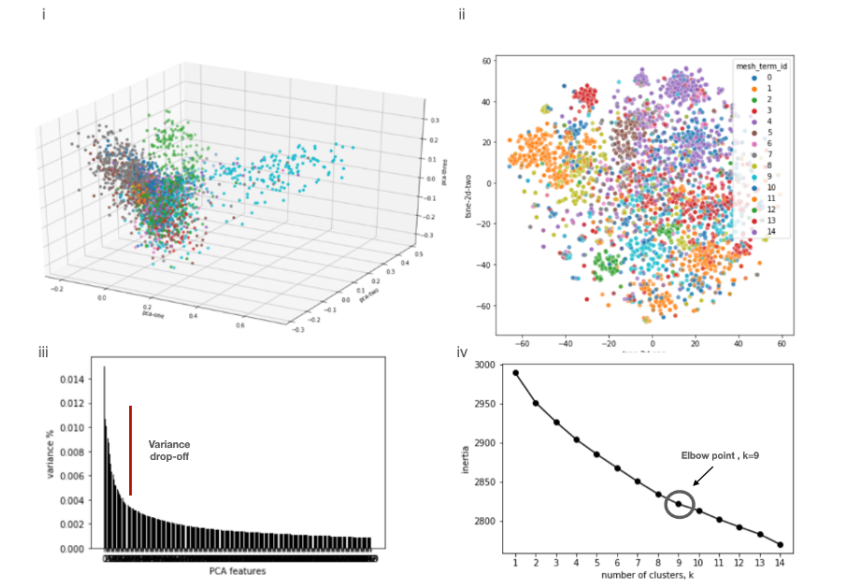

Figure 15: Dimension Reduction for Visualisation Level2

Fig.15 Level 2 of the Mesh Tree Hierarchy:

i PCA , first 3 principal components extracted and displayed on 3D scatter plot. Colours correspond to mesh classes.

ii T-SNE with PCA initialisation. Colours correspond to mesh classes.

iii Scree plot showing variance drop-off after the third component.Cumulative explained variation for 250 principal components: 0.485

iv K-means elbow plot to identify optimal cluster number using 1500 Principal components which explain 80% of the variance. The slight inflection point is circled denoting the 'elbow'.

The first 3 principal component coordinates were plotted. From the graph in Fig.14(i) and Fig.15(i) we can see these three components definitely hold a certain amount of information, but clearly not enough to set all of them apart. The first three principal components contribute 3.1% of the total variation in both datasets. (See proportion of variance % in Fig.14(iii) and Fig.15(iii)). This shows that the data captured in the first three principal components is not informative enough to discriminate the categories from one other.

t-SNE is computationally intensive for cases of very high dimensional data such as ours as it scales quadratically in the number of objects N. For this reason

another dimensionality reduction technique was applied before using t-SNE for visualisation. PCA was done and the cut-off of cumulative 80% variation was chosen to retain 1500 principal components for further analysis. t-SNE was then applied to the resultant 1500 principal components which contribute 80% of the variance. From Fig.14(ii) and Fig.15(ii) there are some distinguishable clusters although not clearly separated with an observable 'crowding issue'. There appear to be a lot of points lying somewhere in the middle not really belonging to any cluster. This could be a function of the dimensionality reduction method or it could be a feature of our embedding. Specifically for t-SNE, the parameter values used have a significant influence on the distribution of the data points. The slight variance contribution for each principal component seen in the PCA skree plot (Fig. 14(iii) and Fig.15(iii)) indicates a low correlation between the variables which do not lend themselves to identifying common dimensions that explain a lot of the underlying variance.

Next we used k-means 'elbow' method to select the optimal number of clusters by fitting the model with a range of values for k. Specifically, we fitted 1500 principal components to the k-means algorithm to determine if there is a cluster structure in the data. Grouping data together in similar classes using the k-means model is done by minimizing the within-cluster sum-of-squares or inertia. Ideally the cluster number where the decrease in inertia reduces and levels off can be chosen as the right cluster number for the data. As k-means clustering minimizes within-cluster variances this can also be interpreted as the point where the percentage of variance explained by the clusters becomes marginal giving an angle in the graph. This is know as the elbow method.

The k-means Elbow method (see Fig. 14(iv) and Fig.15(iv)) indicates the percentage of variance explained as a function of the number of clusters. Only a slight elbow is visible supporting the lack of clearly differentiated clusters in the corresponding t-SNE plots. It should be considered that some documents may simply be vague in their context. For example, a specific abstract from one mesh class may be using many terms which are shared by abstracts from another category and thus their vectors might be more similar to each other than to vectors from their own mesh classes.

5.2 Doc2Vec

To further examine the data and gain understanding of how it may be interpreted by machine learning models, we considered tools for exploring our data given the many thousands of dimensions. Here we used embeddings, a mathematical vector representation that captures different features (dimensions). Each abstract was represented as a vector of 200 dimensions and are positioned in the vector space such that similar abstracts in our corpus are located in close proximity to one another. Specifically embeddings for each word in the cleaned text were imported from the Spacy NLP library and averaged for each abstract to create a document embedding [43]. These document vectors were fed into the k-means algorithm for clustering where the number of clusters was set to equal the number of mesh_terms for each corresponding level of the tree. A comparison of clusters and mesh_terms was done to determine the extent of overlap. This was done both quantitatively and qualitatively.

Exploring the data qualitatively was done by visualising the data in a multi-dimensional space. Our 200-dimensional document embeddings were analysed with the interactive Embedding Projector web application tool. t-SNE modelling was selected to render a three dimension projection. The technique finds clusters in the data by giving each data point (abstract/ document) a location and calculating similarity to its nearest neighbours thereby making sure that an embedding preserves the meaning in the data. The points were colour labelled by cluster or mesh_label for comparison (see Fig 16). From the image projections we see patterns that are not identical across areas where colours fail to merge, suggesting the degree of overlap may be lower than anticipated. This is measured quantitatively by AMI in the next section.

From looking at these 3D t-SNE projections of abstract embeddings in vector space we were also able to get a general idea of the 'modellabillity' of our dataset (Fig 16). On close inspection one might suspect that for each mesh class similar documents agglomerate more in Level 2 than Level 1 where the different coloured points appear more dispersed. This may be interpreted as not having sufficient separation of the classes in the multidimensional space for classifiers to resolve.

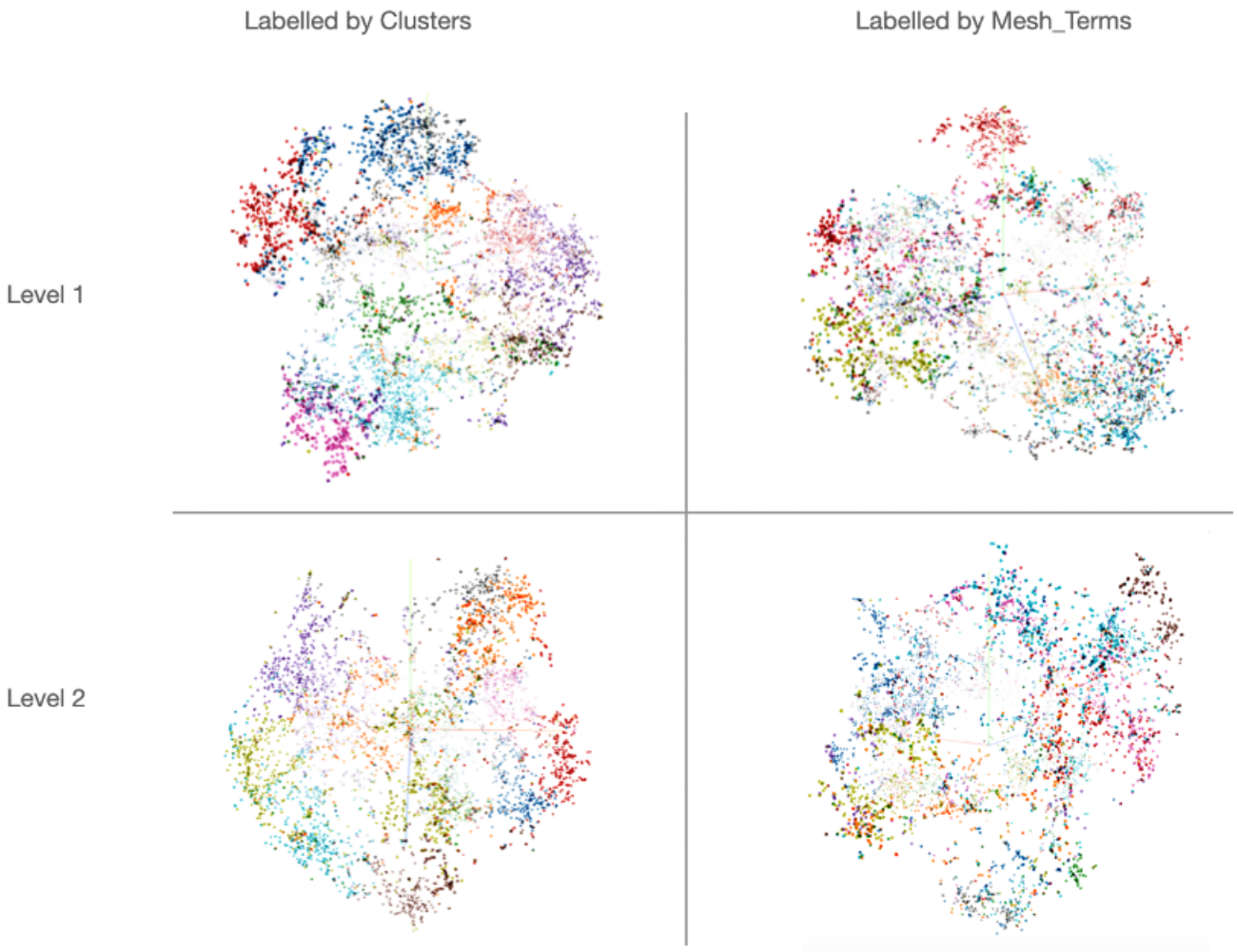

Figure 16: 3-Dimensional TSNE plot (perplexity=25, iterations =1000) courtesy of The Embedding Projector. An open-sourced and integrated TensorFlow platform at projector.tensorflow.org.

5.3 Topic Modelling

Topic modelling was performed using LDA on Tf-idf vectorised clean text and the results visualised with pyLDAvis [44] (see Fig. 17).

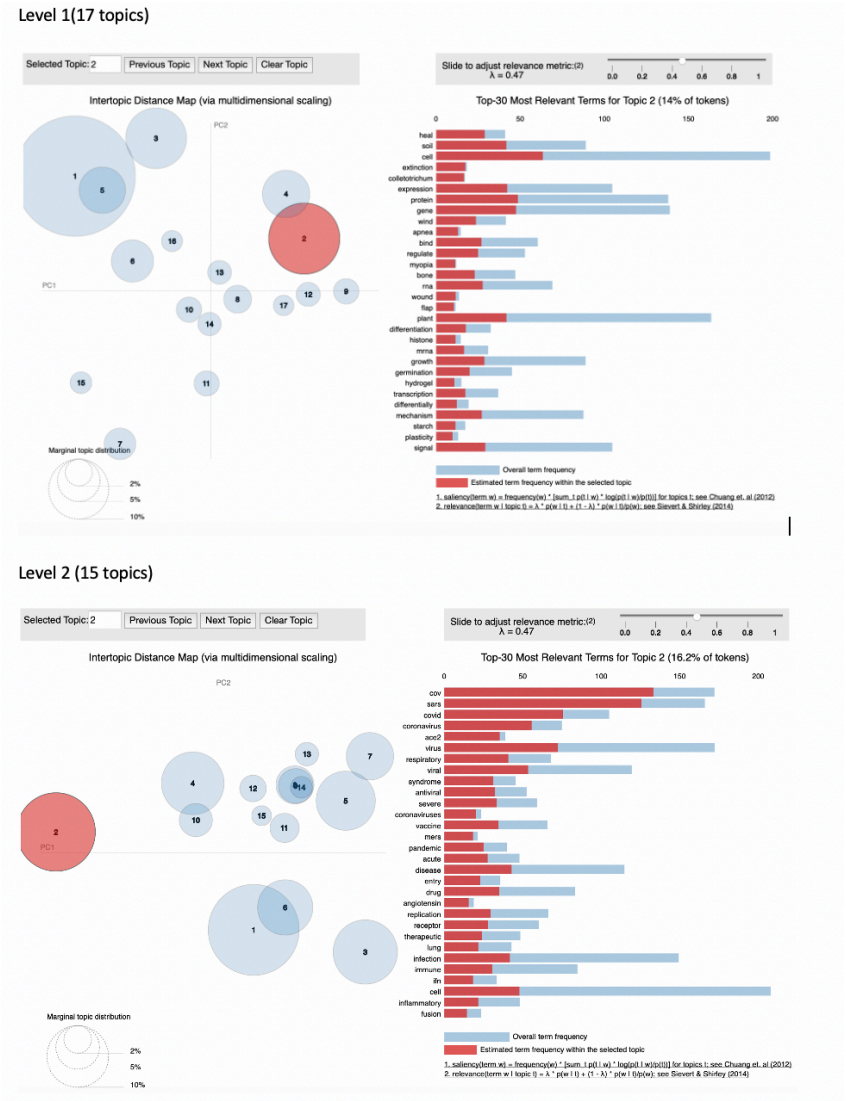

Figure 17: pyLDAvis provides an interface for visualising and interpreting topics.

The LDA algorithm estimates the parameters that define a mixture of topics that the document is about. The topic with the highest probability in the distribution was labeled as the dominant topic (Fig 13 column 5). The distribution of documents by topic label shows that the large majority (greater than 75%) of documents were grouped into 3 predominant topics for both Level 1 (Fig. 18 top left) and Level 2 (Fig. 18 bottom left). This invalidates further supervised learning on topic assigned labels for mesh categorisation given the heavily skewed distribution across labels.

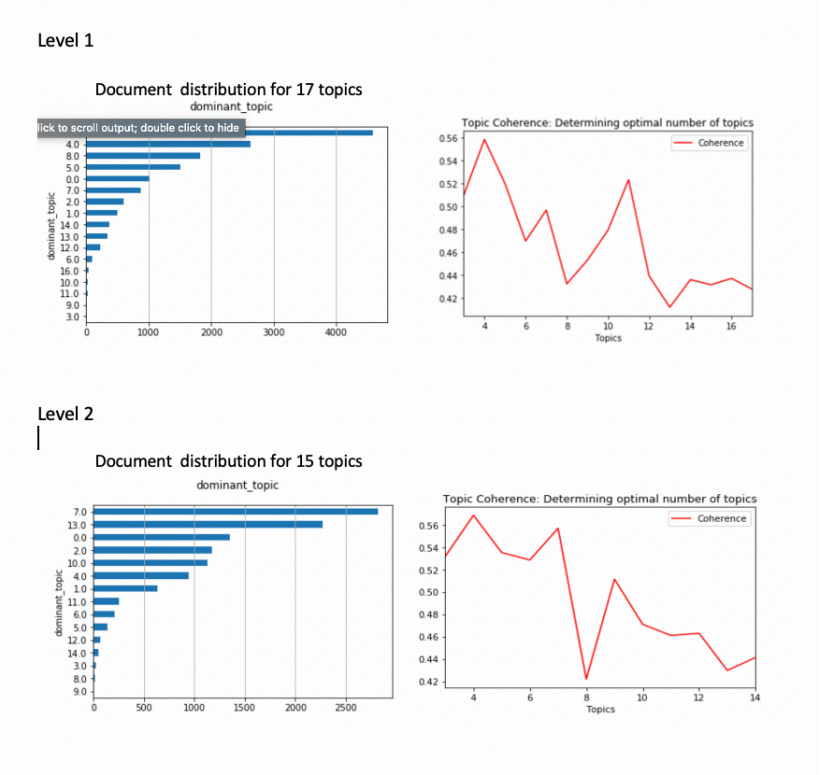

Figure 18: Topic Coherence

Following this, the ideal number of topics given the data was determined using topic coherence scores. The upper plot in Fig.18 shows that for Level 1 of the mesh tree the coherence score fluctuates with the number of topics where the highest reading obtained was for 4 topics. Choosing 11 topics (coherence score = 0.52) may resolve more granular sub-topics. The coherence score for 17 topics was **0.380**. For Level 2 of the mesh tree the lower plot in Fig.18 shows the coherence score decreases abruptly after 7 topics. The coherence score for 15 topics was **0.401**. Where the selected k number of topics equals the number of mesh_terms, the coherence scores are in both cases consistently low relative to fewer topics. For example four topics may be better (more coherent). The low scores overall suggest the topic-related words tend to be semantically unrelated (topics less well-defined).

5.4 Mutual Information

For both k-means clustering and LDA topic modelling , the number of topics or clusters needs to be prespecified. In both instances the number was set to equal the number of mesh_term categories.

AMI is a popular tool for clustering comparisons. Where MI measures the degree of reciprocal matching, AMI is statistically normalised: it is equal to zero when MI is equal to the expected value obtained by chance[52].

The similarity score exists in the range [0,1], where 1 corresponds to identical clusterings and 0 implies maximally dissimilar clusterings. The larger the value, the greater the relationship between the two variables. MI is often used as a general form of a correlation coefficient.

When comparing the assigned clusters or topics with mesh_terms in this study, the scores were closer to zero than one (see Fig.19), although slightly improved for clusters than topics. We tested the null hypothesis being that the mutual information is zero by estimating the p-value as a probability that a random arrangement has a higher score than the true value = a measure of correlation between mesh label and topic/cluster assignments. If the approximated p-value estimate is $p - value = 0.0$, then of all the random arrangements, none had a value higher than that of the true score. The scores reported are therefore considered statistically significant, even though they fall into the lower range. Therefore the topics/clusters are not entirely independent and qualifies as some degree of relation between the 2 sets.

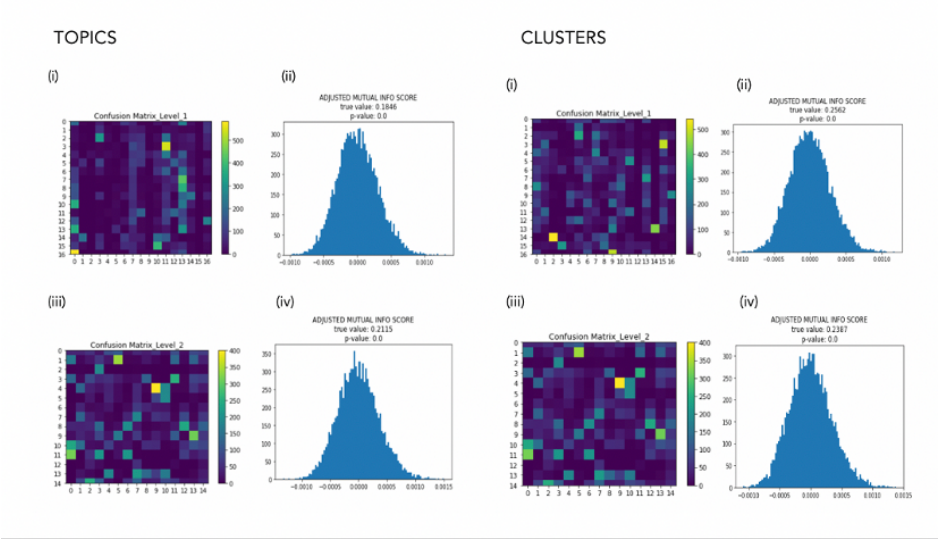

Figure 19: Confusion Matrices and Adjusted Mutual Score for actual Mesh Terms compared with predicted Topics and Clusters

5.5 Multi-Class Classification

5.5.1 Machine Learning Models

The best model construct was determined by the highest mean accuracy value. The graphs in Fig.20 reflect the performances for each of the supervised models over 5 training-validation cycles. The hyperparameters were tuned for each model and accuracy scores reflect best model performance. Chosen hyperparameters can be viewed in the Appendix Table (see Fig.29).

The best classifier for level 1 and level 2 was logistic regression. SVMs are highly robust classifiers that try their best to find interactions between our extracted features, but are not as good as logistic regression. Naive Bayes Classifiers consider the features as independent, resulting in poorer performance than SVMs since it does not take into account any interactions between different features.

The logistic regression models were evaluated further. Fig.21 and Fig.23 show confusion matrices and ROC AUC detailing precision-recall scores for each mesh_term category. The logistic regression model records higher f1-score for some classes relative to others, indicating that some terminology may be more discriminatory for certain classes than for others. For instance the logistic regression model trained on Level 2 mesh classes achieved 70% accuracy (Fig

23) for the test set although struggles to recognise 'Bacterial Physiological Phenomena' (only 99 predicted correctly). By extension this may also explain the higher scores obtained for the more specialised level 2 branch of the mesh tree (third layer of the hierarchical tree structure) when compared to the more generalised groupings of level 1 (second layer of the hierarchical tree structure).

To fully appreciate how the models have learnt to predict mesh classes, we considered the f1 score for each and visualised the individual classes, to get an idea of how similar or dissimilar the data points are to each other. This aids our understanding of why the model made certain predictions, and scores higher for some classes in respect of others. This was done for both tree layers. For Layer 1, clean text was used to generate document word vectors which were fed into the t-SNE algorithm, where for Layer 2 , extracted entities were used as this corresponded with the logistic regression classifiers best performance of 0.726 (Fig. 20).

Embedding Projector was the tool used to translate our high dimensional document vectors into a low dimensional space while preserving the semantic relationships of the inputs [45]. The patterns of difference between the classes, in terms of distribution and spread of documents by t-SNE were visualised for Level 1 (see Fig.22) and Level 2 (see Fig.24) for each respective category. Classes were grouped according to the f1 score reported for each layer (refer to Fig.21 and Fig.23 respectively), from highest to lowest in descending order from left to right. It is evident on close visual inspection that tightly clustered points are positively correlated with f1 score and higher prediction accuracy of the classifier.

The data does not support accuracy higher than 70%. Some possible explanations may be some input features are noisy and non-informative (of no real value in separation of classes), despite doing chi2-feature selection, and/or entity extraction to mitigate against irrelevancy and the 'curse of dimensionality'. Another possibility is that the classifiers are not 'good' enough. Deep learning models were trained next to try improve accuracy performance.

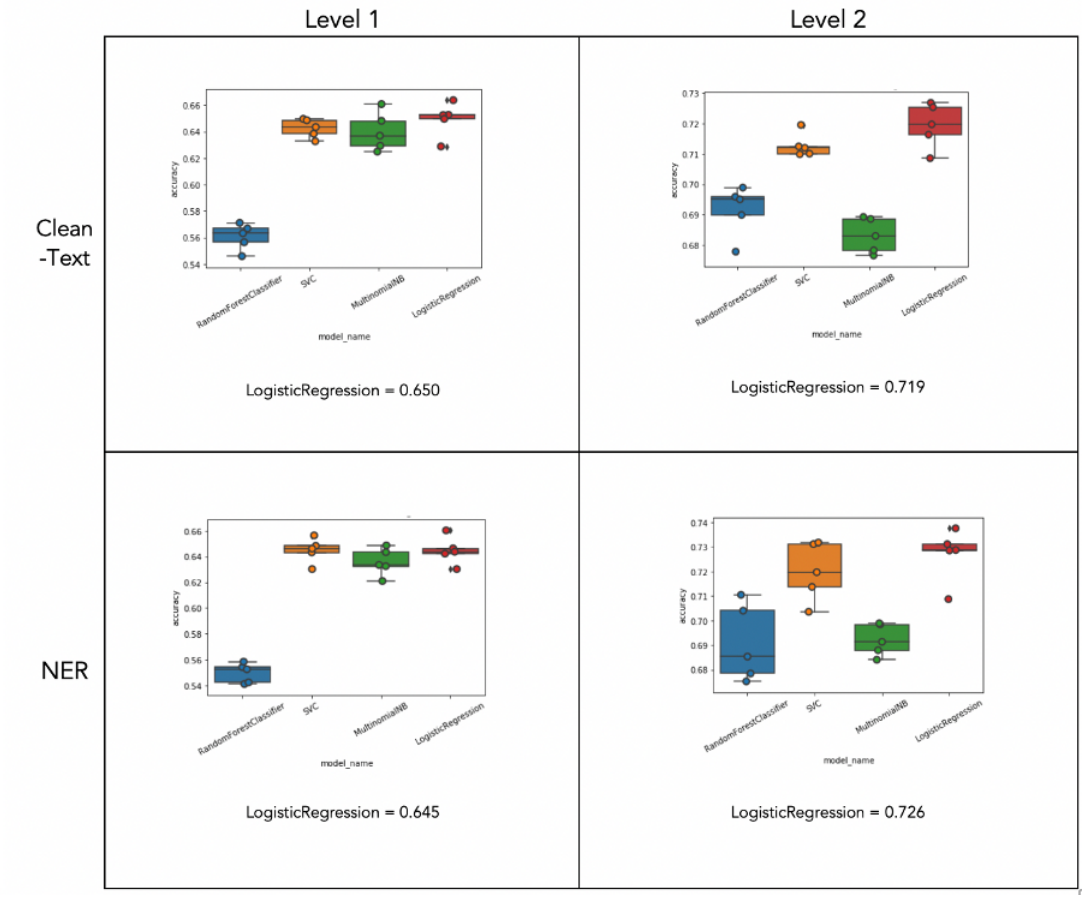

Figure 20: Performance of Supervised Machine Learning Models on selected features for 2 layers of the Mesh Tree Hierarchy

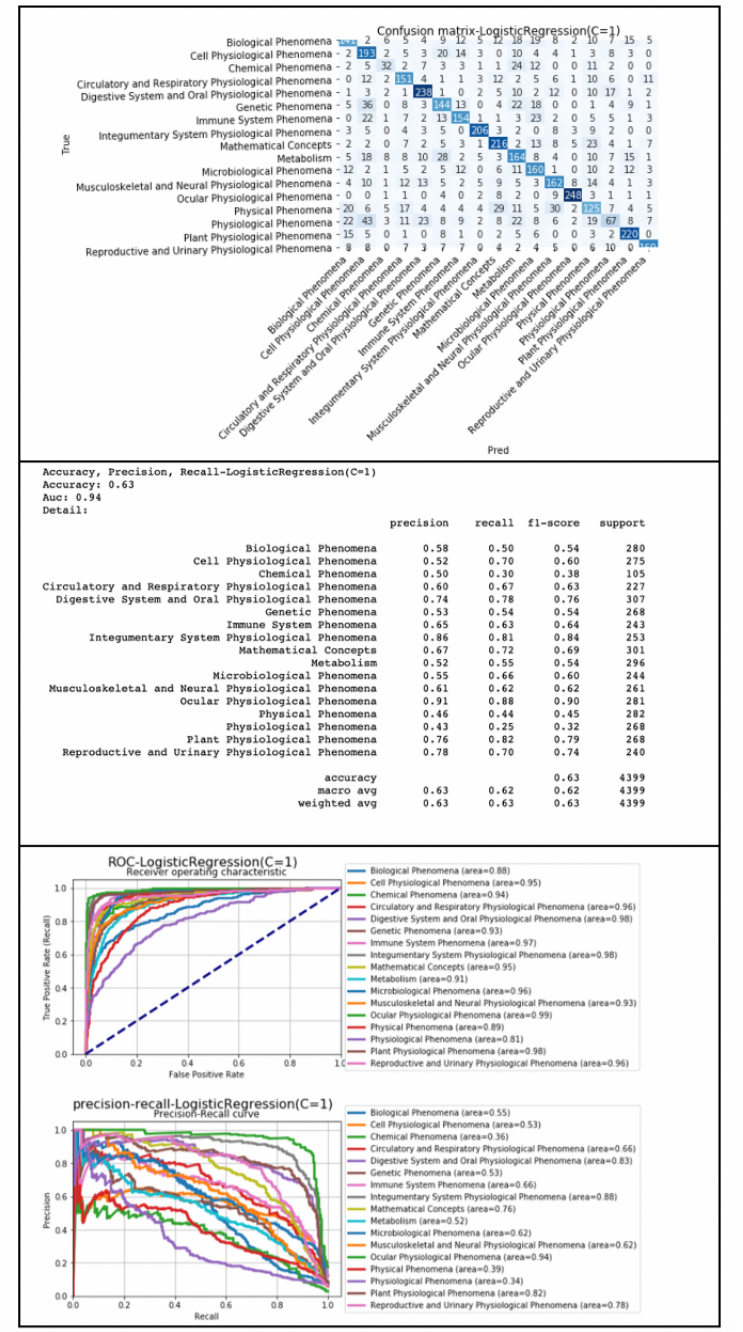

Figure 21: Best Performing Classifier for Level 1: Logistic Regression on clean-text

Figure 22: Level 1: 3-Dimensional t-SNE plots (perplexity=25, iterations=1000) courtesy of The Embedding Projector. Multiple groups in descending order of F1 score from left to right highlighting the distribution of data points for each category

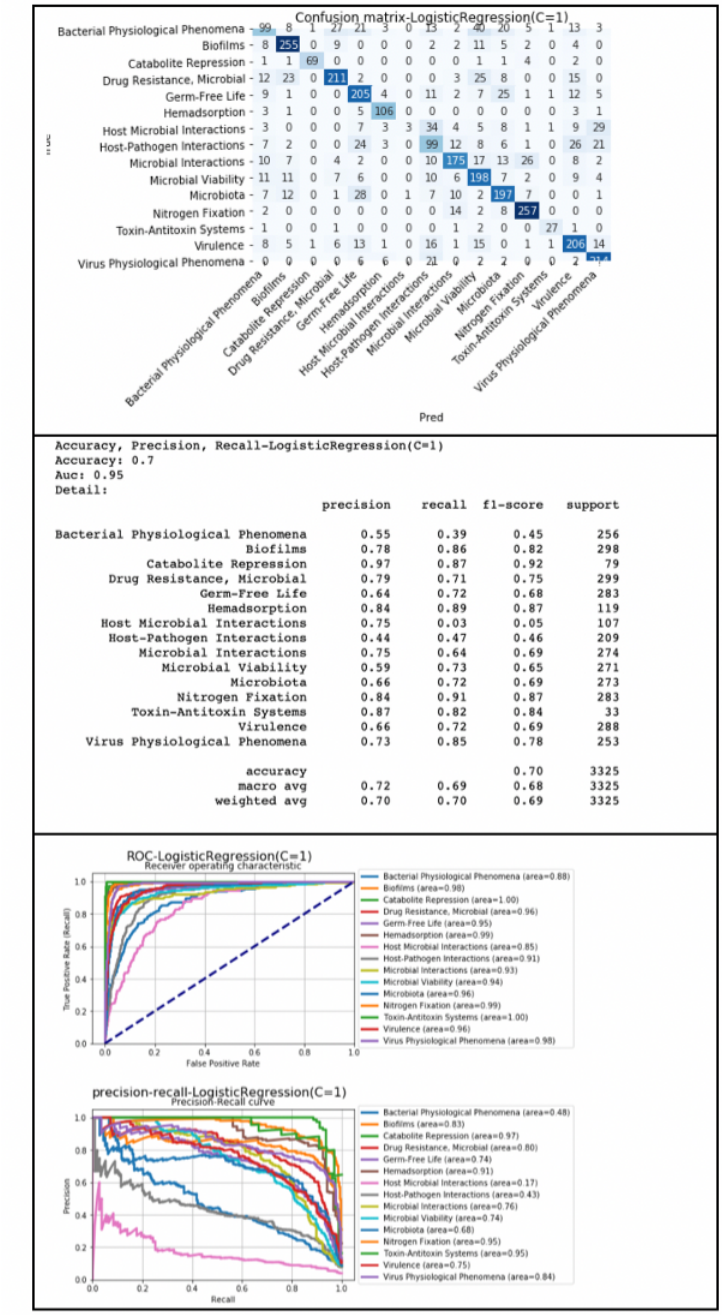

Figure 23: Best Performing Classifier for Level 2: Logistic Regression on extracted entities

Figure 24: Level 2: 3-Dimensional TSNE plots (perplexity=25, iterations=1000) courtesy of The Embedding Projector. Multiple groups in descending order of F1 score from left to right highlighting the distribution of data points for each category

5.5.2 Deep Learning Models

In the previous section, we built a bag-of-words text classifier. The bag-of-words model ignores the order in which words occur. Here we use a 1-dimensional convolutional and recurring neural network that allows us to capture some of the natural ordering that appears in language. Words are represented as embeddings, lower-dimensional vectors of continuous values. We applied dimensionality reduction algorithm t-SNE to visualise a word and its context in a 3 dimensional space after training word embeddings on our corpus (See Fig.25)

The embedding matrix (using either 'in-house' custom trained word embeddings or downloaded 'static' pre-trained word vectors) was used to seed our models predefined embedding layer with parameter weights for the words in our training dataset. Building a model and setting the trainable option to:True allows fine tuning and updating of the embeddings while training. This risks modifying the original word vectors where the learned model may then be less able to generalise on test data. For this reason no further fine tuning of the embedding layer was done and stayed fixed during model training.

The models were trained with a batch size of 256 and a training and validation split of 70 to 30. Results from running the trained models against the test data were recorded for the different configurations in Fig.26.

Again the best performing models for each respective layer were subject to further evaluation (see Fig 27 & 28). LSTM perform overall much better than CNN. The presence of multiple 'gates' in LSTMs allows them to learn long term dependencies in sequences. Modest improvements were observed when working with Spacy embeddings coupled with NER, and SMOTE balancing, although this was not consistent. The best performing model's configuration relied on training the embeddings on cleaned text from a smote-balanced dataset for Level 2 (Accuracy = 0.61). Again, as with the ML models, the improved score compared to Level 1 (Accuracy = 0.55) can be explained by the nature of the scientific jargon used at each level. Specifically as the deeper levels of the MeSH tree become more narrowly specialised, the usuage of more technical terms and less general terminology accounts for the observed difference.

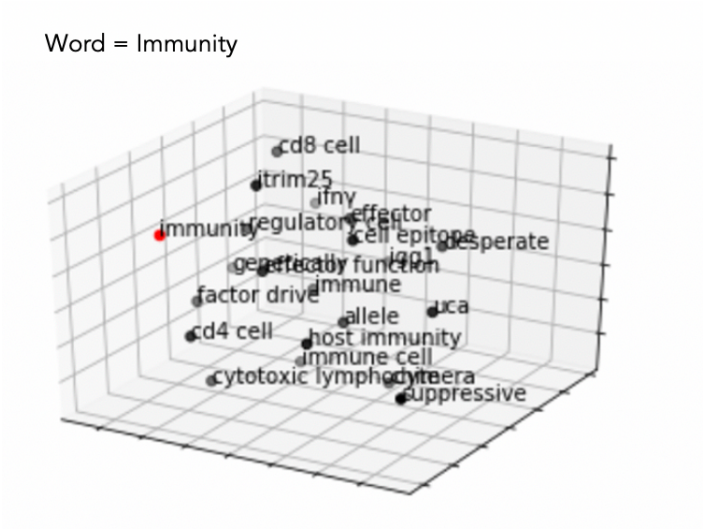

Figure 25: 3D text scatter plot using Tsne to show words similar to 'Immunity' using embeddings trained on our corpus with Word2Vec.

		Level 1		Level 2	
		Accuracy	AUC	Accuracy	AUC
LSTM	Clean_text _ genism_embeddings	0.49	0.89	0.51	0.90
	Clean_text _ trained_embeddings	0.53	0.90	0.6	0.93
	NER _ spacy_embeddings	0.55	0.91	0.55	0.92
	NER_ trained_embeddings	0.54	0.91	0.59	0.92
	Clean_text _trained_embeddings_SMOTE	-	-	**0.61**	**0.93**
	NER _ spacy_embeddings_SMOTE	**0.55**	**0.92**	-	-
CNN	Clean_text _ genism_embeddings	0.36	0.84	0.38	0.84
	Clean_text _ trained_embeddings	0.38	0.84	0.50	0.89
	NER _ spacy_embeddings	0.50	0.90	0.47	0.89
	NER_ trained_embeddings	0.42	0.85	0.50	0.89
	Clean_text _ trained_embeddings_SMOTE	-	-	0.54	0.90
	NER _ spacy_embeddings_SMOTE	0.49	0.89	-	-

Figure 26: Performance Results for Deep Neural Networks

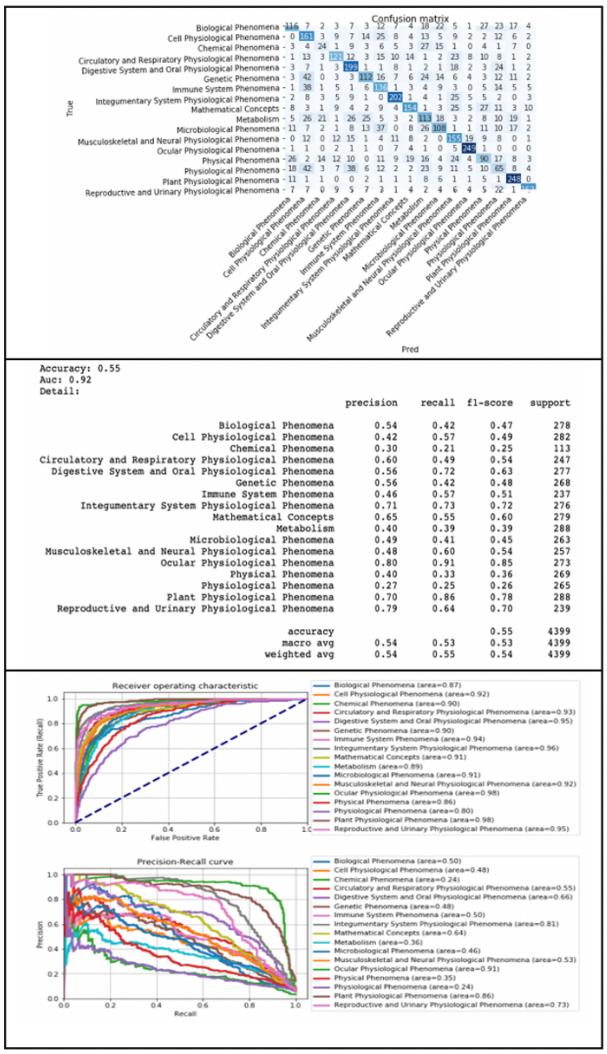

Figure 27: Best Performing Deep Learning Classifier for Level 1: LSTM on extracted entities with Spacy pretrained word embeddings and SMOTE balanced dataset

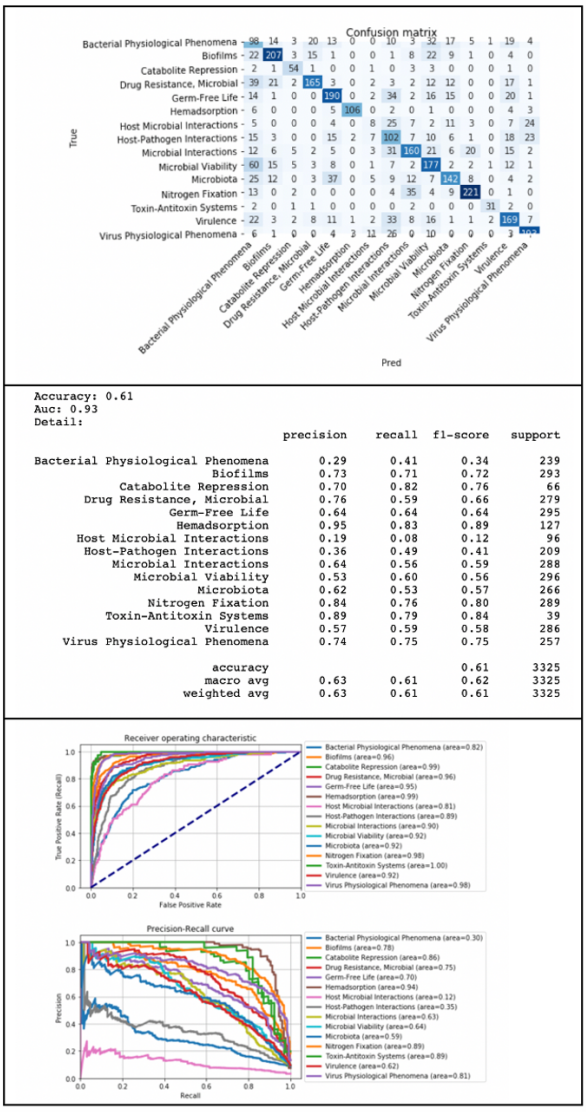

Figure 28: Best Performing Deep Learning Classifier for Level 2: LSTM on cleaned-text with trained word embeddings and SMOTE balanced dataset

6 Conclusions

Automatic MeSH indexing remains a challenging task: the current state of the art performance remains at about 0.6 in F-measure [31, 24]. In this study using the approaches described, we achieved an f1-score of 0.70 with the logistic regression classifier, an appreciable score improvement. For multi-class problems this is considered a good result. When distilled to individual class performance, 2 or 3 classes were weighing down the overall average performance, whilst the remainder achieved good f1-scores.

For this study only the abstracts were used to perform the evaluations, whereas MeSH is based on the entire document. Having full content may allow for more granular separation of labelled documents.

Approaches to combining statistical topic modeling with MeSH terms have been considered. One such study used latent topic vectors corresponding to MeSH terms to create new document representations which improved information retrieval and document classification performance for PubMed abstracts [53].

Classification depends on having documents already indexed with MeSH terms. Further work could focus on unsupervised machine learning algorithms to uncover patterns which could distinguish articles into topics independently of appointed mesh_labels. Topic models may provide an alternative view for better ranking, clustering, and summarizing relevant documents. Since topic models can identify patterns in a corpus , they may reveal insights and emerging themes that are not yet present in MeSH. A topic-based feature representation of documents has been shown to outperform the bag-of-words representation when applied to the task of automatic citation screening. The proposed term-enriched topics were more informative and less ambiguous to systematic reviewers [27].

The introduction of deep pretrained language models in 2018 signals the same shift to transfer learning in NLP that computer vision saw. Bidirectional Encoder Representations from Transformers (BERT) is a novel Transformer model, and has been the state of the art for many kinds of NLP tasks [8]. The introduction of such deep pretrained language models have seen a momentous shift to transfer learning in NLP.

BERT however suffer from a major limitation that hinders their applicability in classification of long sequences. BERT cannot take text longer than the maximum sequence length of 512 'word pieces' (each word subdivided into multiple parts). When BERT is applied to long text tasks, truncating inputs by the maximum sequence length has resulted in decreased performance, since the model cannot learn long-range dependencies of the sequence [36]. The computational complexity scales quadratically with the length, a limitation with unacceptable training time and memory usage [51].

BERTSequenceClassification Model is an off-the-shelf pre-trained BERT model with an additional untrained single linear layer on top for classification. Fine tuning BERT on our specific classification task was consistent with the poor performance reported for long inputs (Accuracy < 0.5 , see github for source code). Continuous improvements dedicated to methods able to tackle this limitation are been explored , and pose enormous potential for many NLP

tasks in the near future [10].

7 Appendix

7.1 Hyperparameters

Models	Hyperparameters
Decision Tree Forest	max_depth=100 ; max_features=1000; min_samples_leaf=10 ; n_estimators=300
SVM	Cost =1; kernel='linear'; gamma=1
Multinomial NB	alpha= 0.5 ; fit_prior=False
Logistic Regression	max_iter = 100 ; C= 1; penalty = 'l2'
CNN	kernel_size: 3 ; pool_steps 2 ; optimizer: 'Adam'
LSTM	dropout=0.2 ; optimizer: 'Adam'
t-SNE	Perplexity=25, max-iterations=1000

Figure 29: Parameters for best Accuracy models

7.2 Research code

Github repository, https://github.com/ssavvi/NLP-and-Data-Science.git

References

[1] Kingma DP Adam. a method for stochastic optimization/dp kingma, j. l. ba. In *International Conference on Learning Representations "ICLR*, pages 1–13, 2015.

[2] Pankaj Agarwal and David B Searls. Can literature analysis identify innovation drivers in drug discovery? *Nature Reviews Drug Discovery*, 8 (11):865–878, 2009.

[3] Varsha Dave Badal, Dustin Wright, Yannis Katsis, Ho-Cheol Kim, Austin D Swafford, Rob Knight, and Chun-Nan Hsu. Challenges in the construction of knowledge bases for human microbiome-disease associations. *Microbiome*, 7(1):1–15, 2019.

[4] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] Jason Brownlee. A gentle introduction to pooling layers for convolutional neural networks, Jul 2019. URL `https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/`.

[6] Jason Brownlee. Best practices for text classification with deep learning, Aug 2020. URL `https://machinelearningmastery.com/best-practices-document-classification-deep-learning/`.

[7] Jason Brownlee. How to use roc curves and precision-recall curves for classification in python, Jan 2021. URL `https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/`.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Ferry Djaja. Multi class text classification with keras and lstm, Jun 2020. URL `https://djajafer.medium.com/multi-class-text-classification-with-keras-and-lstm-4c5525bef592`.

[10] Ahmed Salem Elhady. Bert: Working with long inputs, Jan 2021. URL `https://blog.agolo.com/bert-working-with-long-inputs-866479b9ac7e`.

[11] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[12] Burkhardt Funk, Shiri Sadeh-Sharvit, Ellen E Fitzsimmons-Craft, Mickey Todd Trockel, Grace E Monterubio, Neha J Goel, Katherine N Balantekin, Dawn M Eichen, Rachael E Flatt, Marie-Laure Firebaugh, et al. A framework for applying natural language processing in digital health interventions. *Journal of Medical Internet Research*, 22(2):e13855, 2020.

[13] Sampath Kumar Gajawada. Chi-square test for feature selection in machine learning, Oct 2019. URL https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223.

[14] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[15] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[16] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77: 354–377, 2018.

[17] Casper Hansen. Neural networks: Feedforward and backpropagation explained, Mar 2020. URL https://mlfromscratch.com/neural-networks-explained/#/.

[18] Minlie Huang, Aurélie Névéol, and Zhiyong Lu. Recommending mesh terms for annotating biomedical articles. *Journal of the American Medical Informatics Association*, 18(5):660–667, 2011.

[19] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[20] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[21] Linus Kohl. The similarity of medical texts using the umls ontology, Aug 2020. URL https://medium.com/swlh/similarity-of-medical-texts-using-the-umls-ontology-900bac136f01.

[22] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[23] Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. Adapting svm for natural language learning: A case study involving information extraction. *Natural Language Engineering*, 15(2):241–271, 2009.

[24] Yuqing Mao and Zhiyong Lu. Mesh now: automatic mesh indexing at pubmed scale via learning to rank. *Journal of Biomedical Semantics*, 8(1): 1–9, 2017.

[25] Phayung Meesad, Pudsadee Boonrawd, and Vatinee Nuipian. A chi-square-test for word importance differentiation in text classification. In *Proceedings of International Conference on Information and Electronics Engineering*, pages 110–114, 2011.

[26] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*, 2020.

[27] Yuanhan Mo, Georgios Kontonatsios, and Sophia Ananiadou. Supporting systematic reviews using lda-based document representations. *Systematic Reviews*, 4(1):1–12, 2015.

[28] SPFGH Moen and Tapio Salakoski2 Sophia Ananiadou. Distributional semantics resources for biomedical text processing. *Proceedings of LBM*, pages 39–44, 2013.

[29] Sunil Mohan and Donghui Li. Medmentions: a large biomedical corpus annotated with umls concepts. *arXiv preprint arXiv:1902.09476*, 2019.

[30] James Montantes. 3 reasons to use random forest over a neural network–comparing machine learning versus deep..., Feb 2020. URL `https://towardsdatascience.com/3-reasons-to-use-random-forest-over-a-neural-network-comparing-machine-learning-versus-deep-f9d65a154d89`.

[31] JG Mork, A Jimeno-Yepes, and AR Aronson. Bioasq@ clef. In *CEUR Workshop Proceedings: Aachen, Germany*, 2013.

[32] Mohammed Zeeshan Mulla. Cost, activation, loss function—— neural network—— deep learning. what are these?, May 2020. URL `https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de`.

[33] Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 97–109, 2018.

[34] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. Scispacy: Fast and robust models for biomedical natural language processing. *arXiv preprint arXiv:1902.07669*, 2019.

[35] Nils. Introduction to 1d convolutional neural networks in keras for time sequences, Sep 2018. URL `https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf`.

[36] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE, 2019.

[37] Selva Prabhakaran. Principal component analysis (pca) - better explained, Sep 2020. URL `https://www.machinelearningplus.com/machine-learning/principal-components-analysis-pca-better-explained/`.

[38] Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408, 2015.

[39] Simone Romano, James Bailey, Vinh Nguyen, and Karin Verspoor. Standardized mutual information for clustering comparisons: one step further in adjustment for chance. In *International Conference on Machine Learning*, pages 1143–1151. PMLR, 2014.

[40] Dipanjan Sarkar. *Text Analytics with Python: a practitioner's guide to natural language processing*. Apress, 2019.

[41] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[42] Sethneha. Principal component analysis(pca): Guide to pca, Dec 2020. URL `https://www.analyticsvidhya.com/blog/2020/12/an-end-to-end-comprehensive-guide-for-pca/`.

[43] Shane, 2019. URL `https://www.shanelynn.ie/word-embeddings-in-python-with-spacy-and-gensim/`.

[44] Carson Sievert and Kenneth Shirley. LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3110. URL `https://www.aclweb.org/anthology/W14-3110`.

[45] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*, 2016.

[46] Synced. Applying multinomial naive bayes to nlp problems: A practical explanation, Jul 2017. URL `https://medium.com/syncedreview/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation-4f5271768ebf`.

[47] Dolf Trieschnigg, Piotr Pezik, Vivian Lee, Franciska De Jong, Wessel Kraaij, and Dietrich Rebholz-Schuhmann. Mesh up: effective mesh text classification for improved document retrieval. *Bioinformatics*, 25(11):1412–1418, 2009.

[48] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):1–28, 2015.

[49] Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A Persson, Gerbrand Ceder, and Anubhav Jain. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95–98, 2019.

[50] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[52] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11: 2837–2854, 2010.

[53] Zhiguo Yu, Elmer Bernstam, Trevor Cohen, Byron C Wallace, and Todd R Johnson. Improving the utility of mesh® terms using the topicalmesh representation. *Journal of Biomedical Informatics*, 61:77–86, 2016.

[54] Ning Zhong, Yuefeng Li, and Sheng-Tang Wu. Effective pattern discovery for text mining. *IEEE Transactions on Knowledge and Data Engineering*, 24(1):30–44, 2010.