Comparative Analysis of Neural Networks and General NLP Methods on An Academic Journal Article Database

- BSc Computer Science (Data Science) degree

Final Project, Final Report

(Freda) Xiaoyun Yu

Feb. 2023

Table of Contents

List of Figures

[??update page number??]

List of Tables

1 Introduction

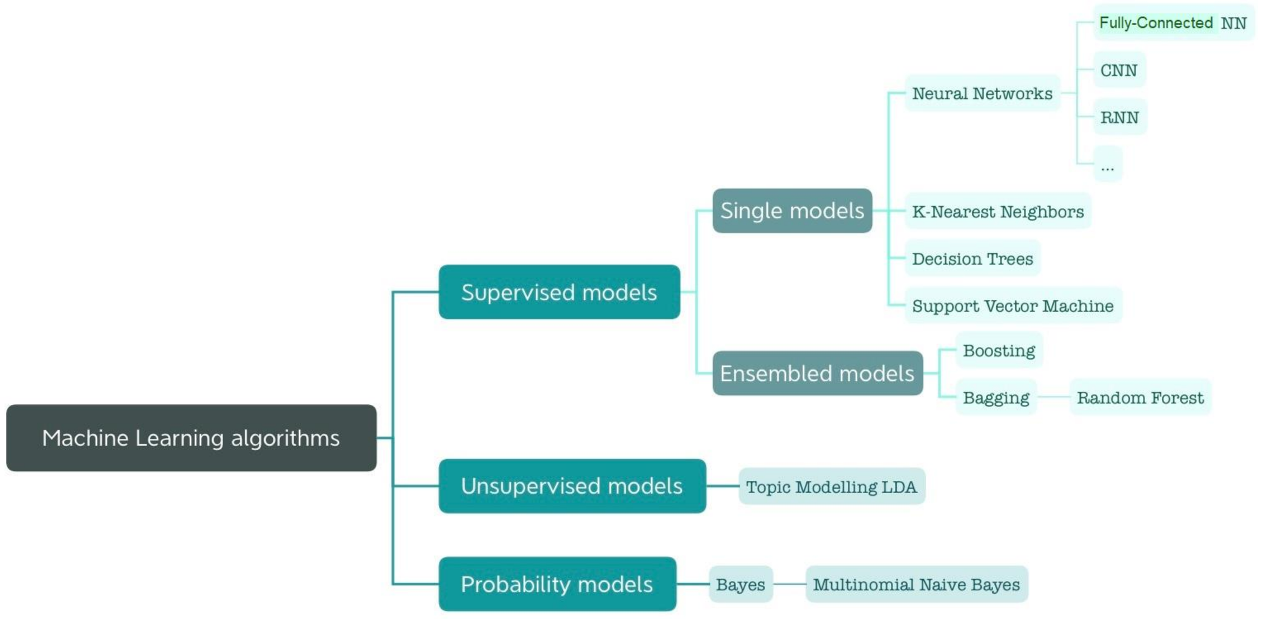
1.1 Template

CM3015-Machine Learning and Neural Networks template.

1.2 Background

The **text classification** field has made significant advancements in recent times, moving beyond traditional NLP vectorizers and classifiers to Deep Neural Network (DNN) models such as BERT, C-LSTM [30], AWD-LSTM [20], ULMFiT [13], and other Transfer Learnings that apply Computer Vision (CV) models to text. Researchers have dedicated considerable efforts to achieving high performance scores on widely used public databases, including DBpedia, IMDB, Penn Treebank [20], and WikiText-2 [20]. However, very few have conducted experiments on topic-targeted databases like **academic journal article databases** [22, 23, 24, 25, 26, 27]. In neural network, there is no single universal model that can perform best for every dataset [37], highlighting the need to fine-tune hyperparameters or use pre-trained models on the academic article database. **The scarcity of prior research on this type of database** has prompted this project.

The goal of this project is to explore text classification on journal articles databases because the number of publications in core academic journals is increasingly critical for anyone in academia [2, 4], including college students, postgraduate doctors, professors, and universities. However, academic journal platforms often **lack subject categories in their metadata** section [36], making it difficult for users with specific requests to search for papers in academic search engines. It also prevents data scientists from implementing a more systematic and intelligent database on these article data. Additionally, previous research on academic articles databases usually only uses abstracts as the data source [25], which goes against the recommendations of the famous librarian Melvil Dewey [3].



(Figure 1.2-1 Machine Learning Algorithms / Models)

Machine learning models are diversified, based on the algorithms implemented (Fig 1.2-1), and can largely be used for classification tasks. There are Neural Network (NN) models which are robust and strong to deal with complex information; Support Vector Machine (SVM) models which excel in non-linear and high-dimensional data; Bayesian models which are stable at multi-class classifications; KNN which uses 'lazy-learning'; Random Forest models which belong to Ensemble Learning…

Therefore, I plan to investigate text classification models for the subject-classification of academic journal articles, with my own collected dataset, and I will use full article body as the data. **I will use multiple Machine Learning (ML) methods to: 1) build Fully-Connected Neural networks (FCNN) and tune hyperparameters; 2) build DNN such as CNN, RNN, LSTM, etc.; 3) compare different vectorizers and different classifiers using more models. I aim to compare the performances of these models, and find the best-performing model(s).** Such models will be the foundation baselines and pre-trained models for this rarely researched 'journal articles classification' task. It might inspire an intelligent article search engine, and the information retrieval and management processes. It may also aid in demonstrating the trend of research emphasis, the shift in focus, and even certain societal problems, as well as the challenges in multidisciplinary disciplines.

1.3 Extension From Template

1.3.1 Requirements of Machine Learning Template

1. Develop a deep learning classification model using Tensorflow and more Python libraries; 2. Aim to find the best-performing model by tuning hyperparameters and using different models; 3. Gather a dataset of a moderate size (mine is 600 items); 4. Replication of high-quality published models (ULMFiT, RNN); 5. Significantly improves the common-sense baseline (my accuracy is 0.73~0.85; baseline is 1/6=0.167); 6. Pre-trained models are adopted (a model from Bilibili, FastAI ULMFiT, AWD-LSTM [20], Bert-for-Task, etc.);

1.3.2 Extension of Machine Learning Template

1. Have collected dataset on my own with a relatively large size (600 items); 2. A novel idea for the dataset (journal articles, which has limited literature review papers); 3. Have used NLP models (vectorizers: TF-IDF, Count Vectorizer, Bigram; classifiers: Multinomial Naïve Bayes, SVM, Random Forest, KNN); 4. Have constructed a feasible CNN model with an accuracy score of 0.633; 5. Have implemented LSTM layers inside my neural network; 6. Have tried to use others' settings as a reference (CNN, RNN, etc.); 7. Comprehensive evaluation methods.

1.3.3 Extension of Previous Course Modules

1. CM3060-Natural Language Processing, midterm: Have fixed errors in data cleansing; have used Machine Learning procedure; used FCNN and tuned hyperparameters; used Deep Neural Networks (multi-layer FCNN, CNN, RNN, etc.); compared all results. 2. CM3015-Machine Learning and Neural Networks, final-term: Have used 12 NLP machine learning models and compared them; tuned hyperparameters based on more controlled settings; have used DNN; have proposed the best model; have tried pre-trained models; used K-Fold; used ROC; used Grid Search.

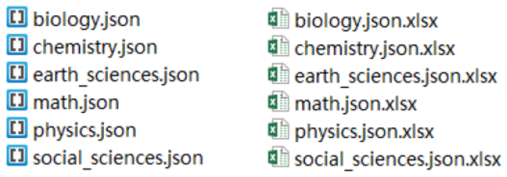
1.4 Selection of Data

The main reason why I do not use only the abstract of each article, but instead **using the full article** is that, the **words used in the abstracts are somewhat misleading** for some categories [3]. For example, in the abstract of a standard biology paper, it may mention some statistical words such as 'p value', 'hypothesis', etc. that may mislead the machine to classify it as statistics - however the statistics here is only the method, not the subject. In fact, most of the papers would contain some of the society descriptions in their abstract, just as an introduction. Thus, it would be misleading only using the abstract. To pursue the accuracy, I would use the full article as the data.

I have collected **6 subjects of articles, 100 for each**, from the EventRegistry.org, using their API https://www.newsapi.ai/documentation/sandbox?tab=searchArticles and sandbox. Since each class has the same amount of data, we do not need class weighting, and this is a balanced dataset. **The commonsense baseline is thus 1/6=0.167.** Using their API, we can get 20+ features about the article, but we only use the full content feature here which follows the pattern of most text classification problems, so we do not have an issue with feature scaling.

After configuring the proper conditions in the sandbox and checking the output articles are correct, I copied their serialisation codes into my Jupyter Notebook and output the desired datasets into JSON files (Fig 1.4-1).

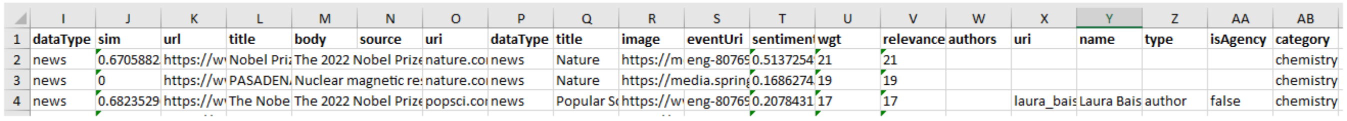
I then slightly tidied up their format, and used an online tool to validate these JSON files (Fig 1.4-1). After that, I uploaded each file into an online tool to convert these JSON files into EXCEL files (Fig 1.4-1).



biology.json	biology.json.xlsx
chemistry.json	chemistry.json.xlsx
earth_sciences.json	earth_sciences.json.xlsx
math.json	math.json.xlsx
physics.json	physics.json.xlsx
social_sciences.json	social_sciences.json.xlsx

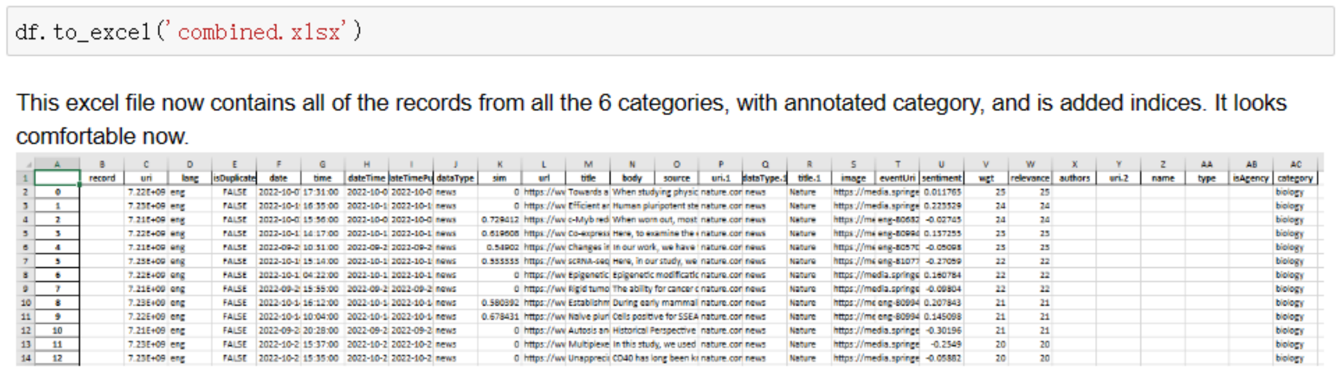
(Fig 1.4-1 Output Data in JSON and XLSX Format)

The converted into EXCEL files contain 27 columns in each. After that, I added a 'category' column for each (Fig 1.4-2), and combined all files into one big EXCEL file named 'combined.xlsx' (Fig 1.4-3) using python codes, which would be the data source file that I would use later. For text classification task, I could only use the article 'body' column in the 'combined.xlsx' as my feature X, and the 'category' column as my target Y.



	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
1	dataType	sim	url	title	body	source	uri	dataType	title	image	eventUri	sentimen	wgt	relevance	authors	uri	name	type	isAgency	category
2	news	0.6705882	https://w	Nobel Pri:	The 2022 Nobel Prize	nature.co	news	Nature	https://m	eng-80769	0.5137254	21	21							chemistry
3	news	0	https://w	PASADEN.	Nuclear magnetic re	nature.co	news	Nature	https://media.spring		0.1686274	19	19							chemistry
4	news	0.6823529	https://w	The Nobe	The 2022 Nobel Prize	popsci.co	news	Popular Sc	https://w	eng-80769	0.2078431	17	17			laura_bais	Laura Bais	author	false	chemistry

(Fig 1.4-2 The Excel File for Each Category with the Annotated Category Column)



```
df.to_excel('combined.xlsx')
```

This excel file now contains all of the records from all the 6 categories, with annotated category, and is added indices. It looks comfortable now.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		record	uri	lang	isDuplicate	date	time	dateTime	lateTimePu	dataType	sim	url	title	body	source
2	0		7.22E+09	eng	FALSE	2022-10-0	17:31:00	2022-10-0	2022-10-0	news	0	https://ww	Towards a	When studying physic	nature.cor
3	1		7.23E+09	eng	FALSE	2022-10-1	16:35:00	2022-10-1	2022-10-1	news	0	https://ww	Efficient ar	Human pluripotent ste	nature.cor
4	2		7.21E+09	eng	FALSE	2022-10-0	15:56:00	2022-10-0	2022-10-0	news	0.729412	https://ww	c-Myb red	When worn out, most	nature.cor
5	3		7.22E+09	eng	FALSE	2022-10-1	14:17:00	2022-10-1	2022-10-1	news	0.619608	https://ww	Co-express	Here, to examine the	nature.cor
6	4		7.21E+09	eng	FALSE	2022-09-2	10:51:00	2022-09-2	2022-09-2	news	0.54902	https://ww	Changes in	In our work, we have	nature.cor
7	5		7.23E+09	eng	FALSE	2022-10-1	15:14:00	2022-10-1	2022-10-1	news	0.555555	https://ww	scRNA-seq	Here, in our study, we	nature.cor
8	6		7.22E+09	eng	FALSE	2022-10-1	04:22:00	2022-10-1	2022-10-1	news	0	https://ww	Epigenetic	Epigenetic modificatic	nature.cor
9	7		7.21E+09	eng	FALSE	2022-09-2	15:55:00	2022-09-2	2022-09-2	news	0	https://ww	Rigid tumo	The ability for cancer c	nature.cor
10	8		7.23E+09	eng	FALSE	2022-10-1	16:12:00	2022-10-1	2022-10-1	news	0.580392	https://ww	Establishm	During early mammal	nature.cor
11	9		7.22E+09	eng	FALSE	2022-10-1	10:04:00	2022-10-1	2022-10-1	news	0.678431	https://ww	Naïve plur	Cells positive for SSEA	nature.cor
12	10		7.21E+09	eng	FALSE	2022-09-2	20:28:00	2022-09-2	2022-09-2	news	0	https://ww	Autosis an	Historical Perspective	nature.cor
13	11		7.23E+09	eng	FALSE	2022-10-2	15:37:00	2022-10-2	2022-10-2	news	0	https://ww	Multiplexe	In this study, we used	nature.cor
14	12		7.23E+09	eng	FALSE	2022-10-2	15:35:00	2022-10-2	2022-10-2	news	0	https://ww	Unapprecia	CD40 has long been kr	nature.cor

	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	uri.1	dataType.1	title.1	image	eventUri	sentiment	wgt	relevance	authors	uri.2	name	type	isAgency	category
2	news		Nature	https://media.springe		0.011765	25	25						biology
3	news		Nature	https://media.springe		0.223529	24	24						biology
4	news		Nature	https://me	eng-80682	-0.02745	24	24						biology
5	news		Nature	https://me	eng-80994	0.137255	25	25						biology
6	news		Nature	https://me	eng-80570	-0.05098	25	25						biology
7	news		Nature	https://me	eng-81077	-0.27059	22	22						biology
8	news		Nature	https://media.springe		0.160784	22	22						biology
9	news		Nature	https://media.springe		-0.09804	22	22						biology
10	news		Nature	https://me	eng-80994	0.207843	21	21						biology
11	news		Nature	https://me	eng-80994	0.145098	21	21						biology
12	news		Nature	https://media.springe		-0.30196	21	21						biology
13	news		Nature	https://media.springe		-0.2549	20	20						biology
14	news		Nature	https://media.springe		-0.05882	20	20						biology

(Fig 1.4-3 Snapshot of the 'combined.xlsx' Data File)

2 Literature Review

2.1 Historical Formation

Classification is not a recently raised issue, but a long existing topic in history. Broadly speaking, classification is to put elements into limited sets, in order to simplify the world.

According to **Daniel Parrochia** (a French philosopher), ancient classification from Plato and Aristotle through the eighteenth century, is 'hierarchical'[14], with limited sets, and is often 'based on a single criterion'. **Ingetraut Dahlberg** [10], a German philosopher who has founded the journal *International Classification* has told us that it was in the 16th century that an Italian philosopher named Mario Nizolio voiced his annoyance that various disciplines may be found under multiple dimensions if shown in a broad plan of the science. In Nizolio's paper '*Antibarbarus philosophicus*', he mentioned several classes to divide science into fields. Even the phylogenetic tree idea proposed by Darwin was inspired by the popular philosophical idea of classification at his time, declared by Parrochia [14].

2.2 Classification in Librarianship

Subject categorization has been a long-standing topic in librarianship, originating in the early years when librarians manually classified books [1, 15, 16].

In Melvil Dewey's book [3] *A Classification and Subject Index, for Cataloguing and Arranging the Books and Pamphlets of a Library* which was written in 1876, he says that 'the library is first divided into nine special libraries which are called Classes. These Classes are Philosophy, Theology, etc., and are numbered with the nine digits . . . and each one is separated again . . . 513 is the 3d Section (Geometry) of the 1st Division (Mathematics) of the 5th Class (Natural Science)'. He has classified books into trees, and the classes under the branch of 'Natural Science' are just what we want to categorize all our articles into. Specifically, they are: mathematics, astronomy, physics, chemistry, geology, paleontology, biology, botany, zoology. I somehow don't agree with his classes, since paleontology, botany and zoology obviously belong to biology. However, the detailed subclasses are the keywords which I should notify in my classification models. What's more, he mentions that the **headings for the books are not reliable enough to classify books**, since 'brevity has been secured in many cases at the sacrifice of exactness'[3]. Thus, if we want to classify our articles, it is sometimes not reliable enough only based on their headings. He even mentions that if the content and the form are in different subjects, such as Mathematical History, then we should classify it into Maths, instead of History, since the content is the crucial one. Thus, **I am considering not to list 'history' as one of the subjects** that I want to classify, or, each article may have multiple tags, which is also a further step from what I want to build.

Mentioned by Parrochia [14], an Indian librarian called **Ranganathan** led the trend in classification during 20th century. His 'faceted classification' (also called "Colon classification" or CC) could permit taxonomists to promptly mix classes into a subject and form a combined version of its class flag, and thus could solve the 'Mathematical History' problem proposed above.

2.3 The Importance of Classification

I have found many books which guide readers in writing articles of the academic field. Svensson [6] has taught us a lot of skills and rules in his book *Academic Journals and Academic Publishing* which emphasises the importance of writing outstanding academic articles. This can demonstrate that my project would be helpful for people who work or study in academia.

Leckie and Buschman[1] have mentioned in their book '*Information Technology in Librarianship*' that children can classify websites into categories, which may inspire neuroscientists and AI scientists to look for new algorithms in classifications.

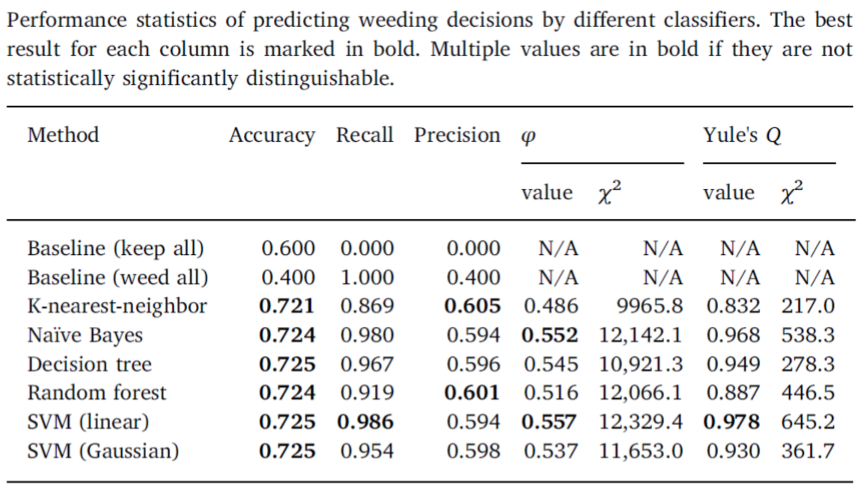
2.4 BERT-based Ensembling Models on Academic Articles

Ambalavanan et al. [17] have stated that text classification of academic articles can assist us in finding a targeting article in a huge database with big amount of articles, although their classification is based on **article quality, rather than the subjects**. They have decided to focus on text classification using **BERT-based ensemble models**. They have developed a unique cascading ensemble that was comparable to the step-by-step screening method used to create the gold standard. With a single integrated model, which they refer to as the individual task learner, they have compared the performance of the ensembles (ITL). They have performed trials using a manually annotated dataset of about 49 K MEDLINE abstracts, and utilised SciBERT, a variation of BERT pre-trained on scientific literature. **They have only used the title and summary** for each article, which was less advanced than mine I assume, according to Dewey's addresses [3]. However their dataset is huge, with 49,028 unique articles. They overall got high recalls (about 0.86+, which means collected fully) but low precisions (about 0.6, which means wrong classification). Their conclusion is: 'while the cascading ensemble achieved the highest precision and the highest F-measure, the single integrated model achieved the highest recall'. I cannot find their code repository.

2.5 Multiple NLP Methods for Binary Classification on Academic Articles

Wagstaff and Liu [15] have taught us in their paper '*Automated Classification to Improve the Efficiency of Weeding Library Collections*' about **the five classifiers in machine learning** especially **for weeding process** (remove outdated items) in library management, which is a **binary classification**. For each classifier, they have filtered the best parameter group. Although their topic is different from mine, I can take consideration of the different performances of these classifiers. From Wagstaff and Liu's paper, we know that there are significant difference between human weeding and AI weeding procedure for library management. Therefore we know that **there are indeed certain tasks in managing articles that must be done by human beings**, and AI certainly cannot replace the complete job of a librarian, or an editor. Their results are listed in Table 2.5-1. They have even considered Chi Square evaluation metric, which is what I should learn from. However, I cannot find their code repository.

Table 2.5-1 Performances of Weeding Classifiers by Wagstaff and Liu

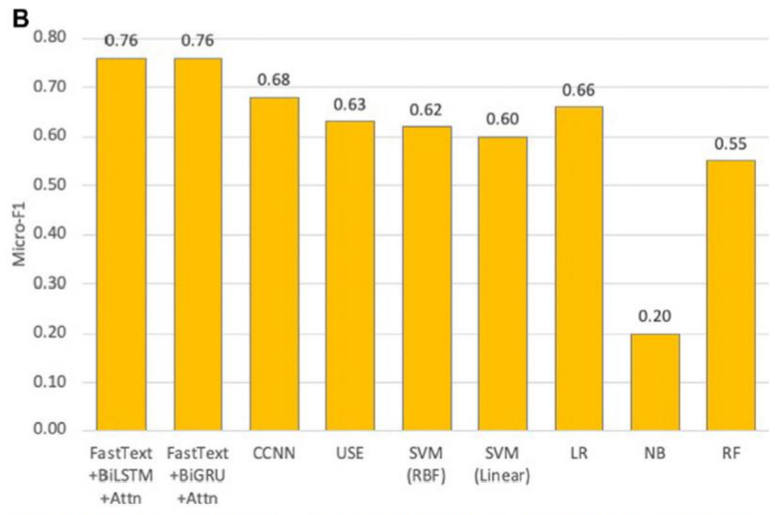


Performance statistics of predicting weeding decisions by different classifiers. The best result for each column is marked in bold. Multiple values are in bold if they are not statistically significantly distinguishable.

Method	Accuracy	Recall	Precision	φ		Yule's Q	
				value	χ^2	value	χ^2
Baseline (keep all)	0.600	0.000	0.000	N/A	N/A	N/A	N/A
Baseline (weed all)	0.400	1.000	0.400	N/A	N/A	N/A	N/A
K-nearest-neighbor	**0.721**	0.869	**0.605**	0.486	9965.8	0.832	217.0
Naïve Bayes	**0.724**	0.980	0.594	**0.552**	12,142.1	0.968	538.3
Decision tree	**0.725**	0.967	0.596	0.545	10,921.3	0.949	278.3
Random forest	**0.724**	0.919	**0.601**	0.516	12,066.1	0.887	446.5
SVM (linear)	**0.725**	**0.986**	0.594	**0.557**	12,329.4	**0.978**	645.2
SVM (Gaussian)	**0.725**	0.954	0.598	0.537	11,653.0	0.930	361.7

2.6 DANN, FastText, LSTM, GloVe, Attn

Kandimalla, Rohatg et al. have written a crucial paper [18] called '*Large Scale Subject Category Classification of Scholarly Papers with Deep Attentive Neural Networks*' which has just the same focus as mine. However, **they only use the abstracts instead of the whole paper** which is a pity. The data size 9,000,000 is huge, and the source of data is Web of Science (WoS) in the year 2015. They have used over 100 categories which is a huge number, and therefore, they have proposed a Deep Attentive Neural Network (DANN) to classify papers in two layers ('Two-Level Classifier'). The proposed network consists of **two Bidirectional RNN followed by an attention layer.** They have just used unigram, not n-grams. **Their best model 'achieves micro-F1 measure of 0.76 with F1 of individual subject categories ranging from 0.50 to 0.95'**. Notice that the **0.76 of F1 is not very high**. Their core findings are: (1) FastText + BiGRU + Attn and FastText+BiLSTM + Attn achieve the highest micro-F1 of 0.76; GloVe + BiLSTM + Attn (micro-F1 = 0.75); GloVe + BiGRU + Attn (micro-F1 = 0.74); FastText + LSTM + Attn (micro-F1 = 0.75); FastText + GRU + Attn (micro-F1 = 0.74); (2) Retraining FastText and GloVe improves the performance; (3) Character-level embedding models often perform worse than word-level embedding models; (4) The best machine learning model (LR) is outperformed by the best DANN model by roughly 10%. They have also acknowledged that the document representation is a restriction - the sequential information is ignored by the BoW model. The Fig 2.6-1 shows their results, which uses a visually direct barplot with a full scale starting from zero that will show us the true values. However, I cannot find their codes.



(Fig 2.6-1 Micro-F1's of the Best DANN Models that Classify Abstracts into 81 SCs, Compared with Baseline by Kandimalla et al.)

2.7 DNN on Smaller Chunks of Discourses

Technically speaking, Lee & Han et al. [19] have explored some of the DNN (Deep Neural Networks) methods on smaller chunks of discourses. Their evaluation scores which are around 0.86 for accuracies of course cannot be compared with my research interest on long passages, but their adjustments on hyperparameters and the choice of other models can inspire me for my task. They have proposed a brilliant way based on Rhetorical Structure Theory (RST) to identify Elementary Discourse Units (EDUs) and their discourse connections. The output is input into a neural network with a tree-like layout that represents the language information, such as the document's organization, discourse roles, and relational types. This model is called 'discourse-aware tree-structured neural networks' by them. They have found that: (1) **words are not as good a unit as sentences in modelling a document**; (2) It is advantageous to compute document embeddings by carefully composing **embeddings of smaller text units** rather than adopting an overview of the document all at once; (3) It is intriguing to see that **LSTMEDUVec performs terribly** even though it looks to be the perfect fusion of LSTM and EDU; (4) Since sentences in the training data are more distinctive than words, **a normal LSTM is anticipated to perform badly with sentences**. They have also discussed about **the gradient-dissappearing problem in the End-to-EndTreeLSTM**, as well as its **space storage** issue. Their results are shown in Table 2.7-1.

They have provided readers with easily-readable graphs to demonstrate their model, starting from the word level, which is visually understandable. However, they have not provided their codes; and in connectedpapers.com, this paper only has 8 citations, which means its impact is very limited; in the main author's Google Scholar homepage, I cannot find this paper either. Another drawback is: for the evaluation metric, they use the accuracy, their reason is that most of their ancestors' works use the accuracy, so they 'have no choice'; but they have not discussed whether their datasets are balanced or not, because the accuracy can hide the polarized performances of an imbalanced dataset. Therefore, it can be seen as a valuable inspiration source, but should not be counted as a reliable baseline.

Table 2.7-1 The accuracies of the baselines with different input text units and composition methods on the test set **by Lee & Han**

Method	Cornell	Stanford	Sarcasm
AvgWordVec	0.7005	0.7281	0.6515
LSTMWordVec	0.7910	0.8861	0.7498
End-to-EndTreeLSTM	0.8700	0.8743	0.7957
AvgEDUVec	0.8560	0.8778	0.8055
LSTMEDUVec	0.5050	0.5888	0.6595
Paragraph Vector	0.7530	0.8248	0.7392
DaNN	0.8755	0.8906	0.8101

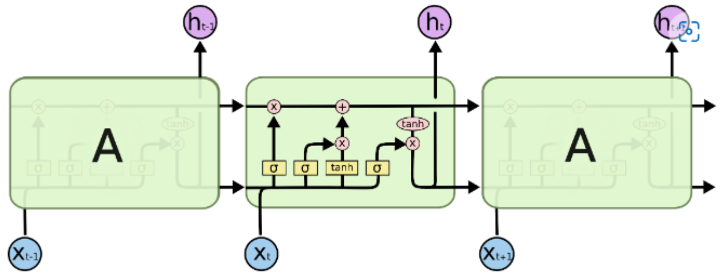
DaNN: Discourse-aware Neural Network.

2.8 ULMFiT

Howard and Ruder [13] have used the state-of-the-art language model **AWD-LSTM** [20](Merity et al., 2017) and they bring to us a **universal LSTM model fine-tuning for classification (ULMFiT), which is a transfer learning model**, can be used as a pre-trained model for any NLP task. Although it is a conference paper, it has detailed methods and thorough comparisons with other methods. In connectedpapers.com, this paper surprisingly has 2658 citations, which is a heavy impact.

They have used 6 comprehensive datasets: TREC-6; IMDb; Yelp-bi; Yelp-full; AG; Dbpedia. An LSTM is a type of RNN (different from a Feed-Forward Network), which uses previous knowledge in decision; but LSTM is a Long-Short-Term model that can deal with the Long-Term Dependency problem in RNN. The Fig 2.8-1 shows an LSTM [32]. The whole paper includes three phases, where they introduce several skills to improve the performance. Phase 1: General-domain LM pretraining; Phase 2: Target task LM fine-tuning – skills: (1) Discrivinative fine-tuning, (2) Slanted triangular learning rate; Phase 3: Target task classifier fine-tuning – skills: (1) Concat pooling, (2) Gradual unfreezing, (3) BPT3C, (4) Bidirectional language model. In the end, the performance of **their ULMFiT on 6 text classification tasks is significantly better than the existing text classification methods**, and the error rate is reduced by 18-24% when tested on most data sets.

Their evaluaion metric is the validation error rate. The whole paper is full of comparison tables in which they have considered various cases and baselines from both their choices and other people's researches. They have compared other people's models: CoVe, oh-LSTM, Virtual, and ULMFiT by McCann et al.; Char-level CNN, CNN, DPCNN, ULMFiT, and ULMFiT by Johnson and Zhang. They have not only provided their codes, but have even built a library called 'FastAI' in Github which has a great amount of users, from which I can borrow some of their hyperparameter settings. However the pity is that, many issues in Github have mentioned the FastAI's slow operation which I have also met in a non-GPU computer.



(Fig 2.8-1 The Repeating Module in an LSTM Contains Four Interacting Layers by Olah)

2.9 AWD-LSTM and DropConnect Techniques

Though the performance of RNN in language model tasks is outstaning, the cyclic connection of RNN is prone to overfitting. The paper named '*Regularizing and optimizing LSTM language models'* proposed by **Merity et al.** [20] is a well-known research centerd on this point, by mentioning many technologies to solve RNN overfitting. This paper is the foundation for many later discoveries. The thesis is to **introduce various regularization techniques in LSTM** to improve the generalization ability of the model and improve the performance of the language model. They use 2 datasets: Penn Treebank; WikiText-2. Their evaluation metric is the word level perplexity. The authors point out that using a fixed-length time-based backpropagation algorithm (BPTT) is inefficient, so **'Variable length backpropagation sequences'** are proposed. They also use a method called '**DropConnect' to subtitute the Dropout** in regularization, since Dropouts will weaken the long-distance ability. Other than DropConnects, Merity et al. do not use SGD here, but investigate **ASGD (averaged SGD)** and find that ASGD has a better effect. They have also investigated other regularization strategies, such as variable BPTT length. They finally study the use of a 'neural cache' in their model, and find that it can further improve the model performance.

The authors have introduced several concepts throughout the work by employing accurate mathematical equations and pseudocodes, and describing them in straightforward language. It is no wonder this paper can be used as a textbook-like foundation for other papers. For the PennTreebank dataset, they have compared 21 models; for the WikiText-2 dataset, they have compared 6 datasets. The results listed in tables are copmrehensive and clear, from which I learn that I need to list both the validation performance and the test performance. They have a Github code repository which is the pre-trained model they propose, for which I have attempted to run in my computer, though they need a GPU to run the codes (they need CUDA package).

2.10 Philosophical Reflections

Back to philosophy again, if we want to refine the classification methods, we need to know the limitations of the idea 'classification'. Parrochia [14] in the 'Internet Encyclopedia of Philosophy' mentions **two instabilities for classifications**. (1) 'Intrinsic instability' is related to the majority of strategies (separations, calculations and so forward) that are utilized to classify objects; (2) 'Extrinsic instability' is associated to the truth that our information is constantly changing, so the definitions of objects (or properties of the objects) are advancing. In my project, I will mainly deal with 'intrinsic instability' technically, and attempt to discuss broadly about some of the 'extrinsic instabilities'.

3 Design

3.1 Domains and Users

As what has been discussed and verified in Introduction and Literature Review sections, this project can be used to: (1) help real journal platforms in academia classify newly received articles and simplify the procedure reviewing new articles, (2) stimulate an interdisciplinary collaboration between these platforms and data science platforms to add comprehensive metadata (as mentioned in Introduction) [36] for information retrieval, (3) boost the development of computer-based technologies to handle part of the librarian's technical jobs [15, 16], and (4) provide inspirations to students and teachers in academia [5, 9, 22, 23]. (5) My models can be applied more widely as pre-trained models in NLP field, since in the 'journal articles classification' field, perhaps my comprehensive discoveries would be a pioneer, compared with limited previous works [22, 23, 24, 25, 26, 27]; and my models if well-tuned, can be applied to more NLP tasks, such as sentiment classification, or even image classification, based on the concept of Transfer Learning [13].

3.2 Timetable

The file for the timetable: https://drive.google.com/file/d/1BreFfxayTG5MiKJAptcTda2ogAQvkBIz/view?usp=sharing Alternatively, see the files in: https://github.com/FredaXYu/FinalProject/tree/main/time_management. A snapshot is in the Fig 3.2-2 in the Appendix.

For milestones, crucial tasks and dates, please see the Appendix Fig 3.2-1.

3.3 Database (Continue with Section 1.4)

3.3.1 For My Models

After having that raw EXCEL file, I need to select the core features, and store them in proper data types. After data cleansing and preprocessing, I have got diversified forms of the same source X_feature and Y_target, prepared for any further processing. Then, I converted them into Keras-recognised data type, and splitted them into training set and test set (Chollet [12] Step 4, Ch. 4.5). The original version and cleansed version for X and Y were both stored (see Fig 4.3-1 in Appendix for the transformation codes). For easier retrieval later, I output my data into 600 .txt files from the name '000.txt' to '599.txt' structured into folders named with numbers '0' to '5', representing 6 categories, with 100 files in each (see my Github '[CODES]ipynb_files/journal_articles' folder).

3.3.2 For Pre-Trained Models in Literature

To use any pre-trained models that predecessors have developed, I had to adjust my data structure to their accepted input data format. To use the 'Bert for Task' pre-trained model which was developed by jiangxinyang227 in Github[38], I firstly added the target label with the separation sign '/<SEP>' in the end of each file; then splitted each of the six categories into train-validation-test sets, with the file numbers of 60, 20, 20, respectively. Then I combined the training files in all categories into one train.txt file (the same for validation and test file). In the end I had three files. This ensured that the ratio of files in each category was balanced and complete.

3.4 Key Technologies and Methods

3.4.1 Text Classification Overview

Text Classification needs a small number of classes, a large set of example documents with document content and class tag, and the input documents data that we care about; the output should be a model (Algorithm 3.4.1-1).

Algorithm 3.4.1-1: Text Classification General Idea

Inputs: A document *d*; a set of classes *C = {c1, c2, …ci}*; a set of example documents, each with document *d* and class *c*: *(d1, c1), (d2, c2), …(dj, cj)*

Output: A classifier *Y*: *d→c*

3.4.2 Machine Learning Workflow Mentioned by Chollet

1. Defining the problem and assembling a dataset; 2. Choosing a measure of success: Accuracy? Precision? Recall? (I will mainly use accuracy); ROC? 3. Deciding on an evaluation protocol (see the 'Evaluation' section below); 4. Preparing data; 5. Developing a model that does better than a baseline; 6. Scaling up: developing a model that overfits; 7. Regularizing the model and tuning hyperparameters. [12]

3.4.3 Controlled Experiment

It is useful when tuning the hyperparameters in FCNN. A controlled experiment contains 3 types of variables: indepenedent variable (the hypermarameter I am studying); dependent variable (the performance, the scores); controlled variables (all of the other hyperparameters and conditions).

3.4.4 Neural Network

A neural network is a computational model consisting of a large number of neurons directly connected to each other. There are many hyperparameters that should be tuned to get a suitable neural network: batch size, activation function in each layer, number of layers, number of neurons in each layer, loss function, regularisation and learning rate, epoch number, etc.

Batch size is how many data to be processed each time. The activation function defines how neurons are determined as ‘active’ and generally is a jumping function; for single-label multi-class classification, we should use softmax [12]. The number of layers will determine the depth of our model. A good NN often needs neuron numbers to decrease at last, otherwise it meets ‘bottleneck’ effect. Loss Function is used for optimisation, where it represents the distance between predicted data and real data and thus should be minimized; Chollet says we should use categorical crossentropy [12]. Reglarisation is to improve the performance on test set by penalising large influences and thus avoid overfitting; there are L1, L2, Dropout. Epoch number is how many iterations we will train; too small will encounter underfitting, while too big risks for overfitting.

Pros: does not need feature engineering; better performance with bigger sample size. Cons: the procedure cannot be expained; the performance under small sample size may be worse than machine learning; model tends to be too complicated.

3.4.4.1 Feed-Forward Neural Networks (FNN)

In FNN, information always goes from previous layer to the later layers and never goes back. There is a Backpropagation algorithm to adjust the current loss function comaring with the final output.

3.4.4.1.1 Fully-Connected Neural Network (FCNN)

FCNN is a basic type of FNN, which is constructed by input layer, hidden layer, and output layer. When layers are enough, a FCNN can be called a ‘Deep Neural Network’. My controlled experiments are largely on FCNN.

3.4.4.1.2 Convolutional Neural Networks (CNN)

Convolution is widely used in image processing. In entity identification, we set a small rule in each convolution matrix, and use it as a filter to examine through the whole image; combining many small rules can achieve a very abstract identification task. This method can be transferred to text classification field, by using word vectors to replace the pixels in the image. The convolutional kernal scans for 2~5 words each time [29].

Pros: shared convolutional core is strong enough to deal with high-dimension data; the classification would be better if we manually select features well and train the weight. Cons: needs tuning; needs big sample size, and may need GPU; the physical meaning is blurred.

3.4.4.2 Feedback Neural Networks

3.4.4.2.1 Recurrent Neural Networks (RNN)

In videos, articles, and voices, the output of the network depends not only on the current input, but also on the outputs at some past stages. In RNN, there are loops that can reference past information, so that it is more sensible to process article data. This is a mechanism that allows more operations among neurons, though the whole procedure still can hardly be conveyed by plain language. Cons: vanishing gradient, gradient explosion in long sequences.

3.4.4.2.2 LSTM, AWD-LSTM

When an input series of data is too long, there may exist the problems of Gradient Vanishing and Gradient Exploding, and thus people have explored variant of methods to improve RNN, including LSTM (Long Short Term Memory). LSTM [32] adds a ‘forgetting gate’ to RNN. AWD-LSTM [20] is a model which has sharpened LSTM, by using many regularisation methods and DropConnect instead of Dropout.

3.4.5 Grid Search

Grid search is a method to brutforcely search for every possible combinations settled down by us, and it returns the best combination. Scikit-learn has a GridSearchCV functionality.

3.4.6 One-Hot Encoding vs. Word Embedding

These are the two main vector representations of texts. One-Hot Encoding uses 0 or 1 to represent the features. In contrast, Word Embedding will bring closer of two similar words. The idea is that, the similar words will appear in the same text context.

3.4.7 Bayesian Classification

Bayes Theorem states: the new probability of a hypothesis after a new evidence coming in, should depends on both the prior knowledge, and the new evidence (Fig 3.4.7-1). It is often used to determine the probability when a hypothesis is true.

$$P(H|E) = \frac{P(H)P(E|H)}{P(E)}$$

(Fig 3.4.7-1 Bayes' Theorem)

It can be extended to be a Bayesian Classifier: we just replace the 'hypothesis' with 'label' which is what we postulate, and we replace 'evidence' with 'feature' which is the thing we've required from the data. Therefore we have Fig 3.4.7-2, which means that, to choose between labels, we compare their posterior probabilities.

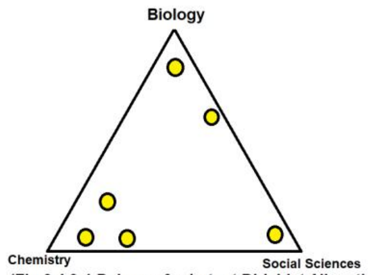
$$\frac{P(L1|\text{features})}{P(L2|\text{features})} = \frac{P(L1)P(\text{features}|L1)}{P(L2)P(\text{features}|L2)}$$

(Fig 3.4.7-2 Bayesian Classifier)

3.4.8 Support Vector Machine (SVM)

Support Vector Machine (SVM) classifies groups by constructing a line (hyperplane) with margins, which is the distance between one group to the other, usually the closest dots. The idea to search for the best-fit line is to maximize the margin. Scikit-learn has SVM algorithm in sklearn.svm package and sklearn.svc.LinearSVC [35].

3.4.9 Topic Modelling: Latent Dirichlet Allocation



(Fig 3.4.9-1 Polygon for Latent Dirichlet Allocation)

Latent Dirichlet Allocation (LDA) is a type of unsupervised learning method. It can give us a settled number of possible aggregations based on some similar features inside of each. Two assumptions of LDA: each topic is a mixture of an underlying set of words; each document is a mixture of a set of topics. For a given number N, LDA draws a polygon with N edges. It decides which data point belongs to which group by observing which angle the data point is the nearest to (Fig 3.4.9-1).

3.5 Evaluation Methods

3.5.1 Underfitting or overfitting?

The key in machine learning models is the 'generalisation', which is to convey a widely suitable model on unseen data, based on our known training data. If we have a too-simple model, the model cannot learn comprehensively on the training set, and thus its description ability is weak. This is underfitting. If we have a strongly written neural network, but train it overly, the training accuracy will be very high, but it describes too much details about this training set but ignores the needs on unseen test set. This is overfitting. To deal with underfitting, we can: (1) add new features; (2) decrease the regularization parameters; (3) use more complex non-linear models such as SVM, Decision Tree, Deep Learning; (4) use ensembling methods to combine low-capacity models. To prevent overfitting, we can: (1) gather more real data; (2) use simulations to generate more data; (3) split dataset into train-validate-test datasets or use K-Fold Cross Validation; (4) data augmentation; (5) use regularization; (6) use dropouts to stop neurons from 'colluding with' each other.

To find the optimal epoch number, we usually use two plots: the Epoch-Loss plot, and the Epoch-Accuracy plot, showing both training set and validation set in each. The loss should be decreasing as training goes on, but if we observe an uprising curve on the validation set, that is the 'overfitting moment' and we should just record the epoch number; then find that epoch number on the Accuracy plot, we should usually find a decreasing trend. That epoch number is what we should settle. We have reasons to try for any 'overfitting moment' until we find the proper epoch number.

3.5.2 Accuracy, Precision, Recall, F1

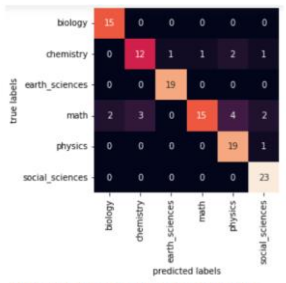
Accuracy = (Correct Predictions)/(Total Predictions). It considers all the categories. As our dataset is balanced, we can use accuracy; and for simplicity, we use accuracy to represent the whole performance of a model.

Precision = (The positive predictions which are correct)/(All the positive predictions) = TP/(TP+FP); high precision is the lack of False Positives (mis-reports). Recall = (The positive predictions which are correct)/(All the true positive cases) = TP/(TP+FN); high recall is the lack of False Negatives (omissions). In multi-category classification, these two metrics help us to form an understanding of our model when it is good at classifying some certain categories while is weak at classifying some others, and thus are more detailed evaluation metrics.

F1 = (2*Precision*Recall)/(Precision+Recall). As it combines precision and recall and gives them the same weight, we can also use it as an overall metric to evaluate the performance on each category.

3.5.3 Confusion Matrix

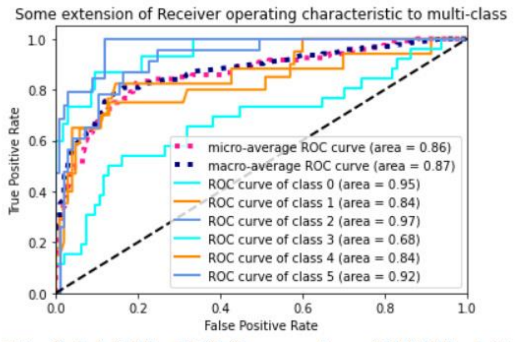
Confusion Matrix is a visualised method to show all the True Positives, True Negatives, False Positives, and False Negatives. As these concepts are relative to each category, in the end we have a square matrix with the row (or column) number equals to the category number. Implemented in Heatmap codes, we can clearly see the overall performance – if the diagonal line is visually obvious, then the performance is good; if all the colours are similar, then that model is bad. Fig 3.5.3-1 shows a strong model's confusion matrix, with 6 categories.



(Fig 3.5.3-1 Confusion Matrix for 6-Classes Classification)

3.5.4 ROC, AUC

ROC (Receiver Operating Characteristic) curve is a plot with each point representing different thresholds, showing the relationship of FPR (False Positive Rate) and TPR (True Positive Rate). As we set the range to be different classes or Micro- or Macro-average, there can be different ROC curves (Fig 3.5.4-1). AUC (Area Under Curve) is a rate to represent the ratio the ROC curve encloses. The bigger the AUC is, the better the model. AUC means 'the probability that the prediction outcome of positive samples is bigger than that of negative samples'. AUC=1 is perfect; AUC=[0.85, 0.95] is outstanding; AUC=[0.7, 0.85] is general; AUC=[0.5, 0.7] is just acceptable; AUC=0.5 is like random guessing; AUC<0.5 is worse than random guessing.



(Fig 3.5.4-1 My ROC Curves for a CNN Model)

4 Implementation

4.1 Data Collection

See the full file of retrieving data here: https://drive.google.com/file/d/17ztt_VkZ0MOn7pQ-x7YcIDwyks2pxahC/view?usp=sharing After installing EventRegistry, I write the Algorithm 4.1-1 to retrieve articles and related information for each category into JSON files:

Algorithm 4.1-1: Information Retrieval from EventRegistry.org Using Their API

```
1: er ← EventRegistry object, with my unique API key
2: q, qStr ← The detailed query generated by their sandbox and functions
3: for each_article in the_query(inputs: er, maxItems=100):
4:   with open(created_empty_json_file):
5:     write to that file with the retrieved each_article
```

Then I convert format into EXCEL files, and summarise them into one file. After that, I read the 'body' (article) column data into x variable, and the category column data into y variable (with integers from 0 to 5 to represent each category).

4.2 Data Cleansing

Steps: Remove unwanted characters; Lowercase and expand contractions; Remove stopwords & lemmatisation. Key algorithms are listed in Algorithm 4.2-1 and 4.2-2:

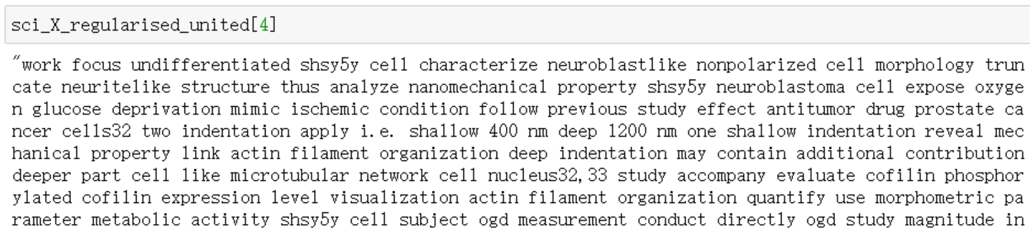
Algorithm 4.2-1: Remove Unwanted Characters

```
Input: a text string text
1: set rules using RegularExpression library, examples: enter, URLs, special chars, html tags, ...
2: for each mentioned above, substitue and replace the old text with the updated version
3: repeat 1~2
Output: the characters-removed version of that string of text
```

Algorithm 4.2-2: Expand Contractions & Lowercase

```
Input: a text string text
1: contractions ← set rules for contractions; new_text ←an empty list to store future list of word strings
2: for each character char in text:
3:   if char is not space:
4:     read the char, and append it into a temporary variable
5:   else if char is space:
6:     join each character to form the word, and lowercase the word
7:     if that word is in the contractions:
8:       append the full version to new_text
9:     else:
10:      just append this word to new_text
11: text ← use space to join these words, then delete spaces at beginning and the end
Output: the new version of that string of text
```

The final cleansed form of one of the article data is shown in Fig 4.2-1:



```
sci_X_regularised_united[4]
```

```
"work focus undifferentiated shsy5y cell characterize neuroblastlike nonpolarized cell morphology trun
cate neuritelike structure thus analyze nanomechanical property shsy5y neuroblastoma cell expose oxyge
n glucose deprivation mimic ischemic condition follow previous study effect antitumor drug prostate ca
ncer cells32 two indentation apply i.e. shallow 400 nm deep 1200 nm one shallow indentation reveal mec
hanical property link actin filament organization deep indentation may contain additional contribution
deeper part cell like microtubular network cell nucleus32,33 study accompany evaluate cofilin phosphor
ylated cofilin expression level visualization actin filament organization quantify use morphometric pa
rameter metabolic activity shsy5y cell subject ogd measurement conduct directly ogd study magnitude in
```

(Fig 4.2-1 The Final Cleansed Form of One of the Article Data)

4.3 Train-Test Splitting, Encoding

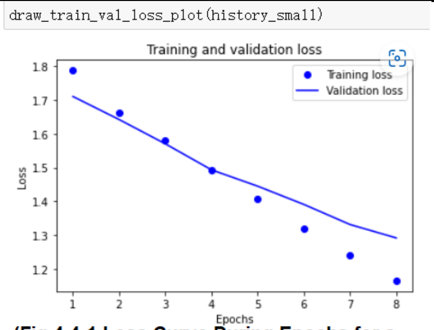
Tensorflow only takes in number data, but we have text data, so we must transform our data, which is known as 'encoding' process, after train-test dataset splitting. (I leave for the splitting of train-validation in writing real models later.) Please see the Fig 4.3-1 in the Appendix for the whole codes of encoding.

4.4 A Small Neural Netwok (Chollet Step5)

Although an underfitting NN is not helpful for developing the most strong model, it is a necessary step described by Chollet [12]. In the Epochs-Loss plot on the validation set (Fig 4.4-1 curve), we see the loss is decreasing without a fluctuation, so it is not yet overfitting. The Epoch-Accuracy curve (Fig 4.4-2) shows an increasing trend, which also means it is still underfitting. We see the total accuracy is 0.50 on test set, which is higher enough than the baseline 0.167 (Fig 4.4-3).

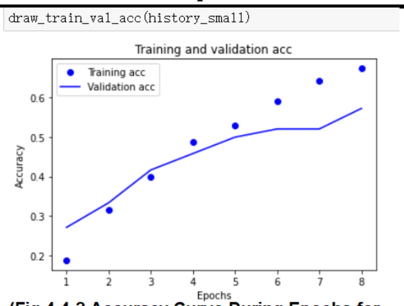
Settings 4.4-1: Small Neural Network which Underfits

All layers are Dense. 3 layers in total. Neuron numbers: 16-16-6
activation function: all are relu, output layer is softmax; optimizer: rmsprop;
loss func: categorical_crossentropy; batch size=64; epochs=8



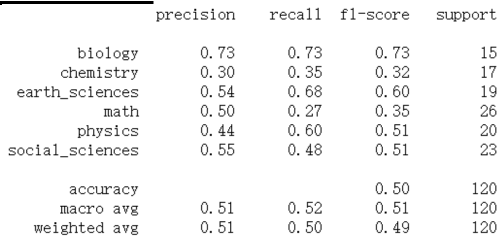
```
draw_train_val_loss_plot(history_small)
```

(Fig 4.4-1 Loss Curve During Epochs for a Small FNN)



```
draw_train_val_acc(history_small)
```

(Fig 4.4-2 Accuracy Curve During Epochs for a Small FNN)



```
                   precision    recall  f1-score   support

        biology       0.73      0.73      0.73        15
      chemistry       0.30      0.35      0.32        17
 earth_sciences       0.54      0.68      0.60        19
           math       0.50      0.27      0.35        26
        physics       0.44      0.60      0.51        20
social_sciences       0.55      0.48      0.51        23

       accuracy                           0.50       120
      macro avg       0.51      0.52      0.51       120
   weighted avg       0.51      0.50      0.49       120
```

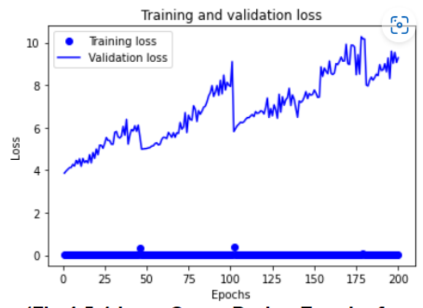
(Fig 4.4-3 Performance Summary of Each Category for a Small FNN)

4.5 An Overfitting Neural Network (Chollet Step6)

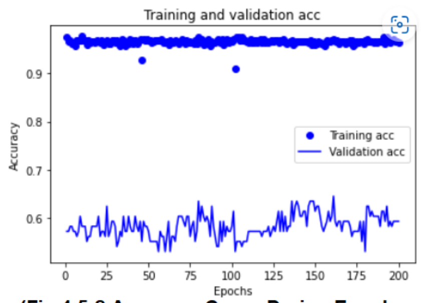
The Settings 4.5-1 shows my overfitting neural network hyperparameters. From Fig 4.5-1 the Epoch-Loss plot, the loss is even increasing, so it must be overfitting. From Fig 4.5-2 we see the accuracy is fluctuating, so it must be overfitting. Although the total accuracy is 0.61>baseline 0.167 which is satisfying, we need to use a smaller epoch number to prevent overfitting.

Settings 4.5-1: Bigger Neural Network which Overfits

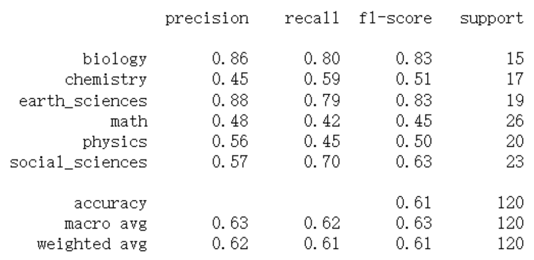
All layers are Dense. 6 layers in total. Neuron numbers: 16-128-256-256-16-6
activation function: all are relu, output layer is softmax; optimizer: rmsprop;
loss func: categorical_crossentropy; batch size=64; epochs=200



(Fig 4.5-1 Loss Curve During Epochs for an Overfitting FNN)



(Fig 4.5-2 Accuracy Curve During Epochs for an Overfitting FNN)



```
                precision    recall  f1-score   support

       biology       0.86      0.80      0.83        15
     chemistry       0.45      0.59      0.51        17
earth_sciences       0.88      0.79      0.83        19
          math       0.48      0.42      0.45        26
       physics       0.56      0.45      0.50        20
social_sciences      0.57      0.70      0.63        23

      accuracy                           0.61       120
     macro avg       0.63      0.62      0.63       120
  weighted avg       0.62      0.61      0.61       120
```

(Fig 4.5-3 Performance Summary of Each Category for an Overfitting FNN)

4.6 Controlled Experiments for Hyperparameters

Much effort has been put on the experiment on each hyperparameter. Till now, I have collected hundreds of experiment records on hyperparameters: batch size, number of layers, activation functions, neuron numbers, optimizers, regularisation. See my complete experiment records here: https://github.com/FredaXYu/FinalProject/tree/main/%5BIMPORTANT%5Dexperiment_records

To demonstrate my designed functions for controlled experiments, I list in Algorithm 4.6-1 a function which experiments on the activation functions as an example. That function can be implemented also for other hyperparameters, since the input parameters contain many variables which are available to be tuned. The real controlling part is when calling this function, where I have ensured that only the *act_list* is the changing part, while other parameters are fixed.

Algorithm 4.6-1: Controlled Experiment for Activation Functions

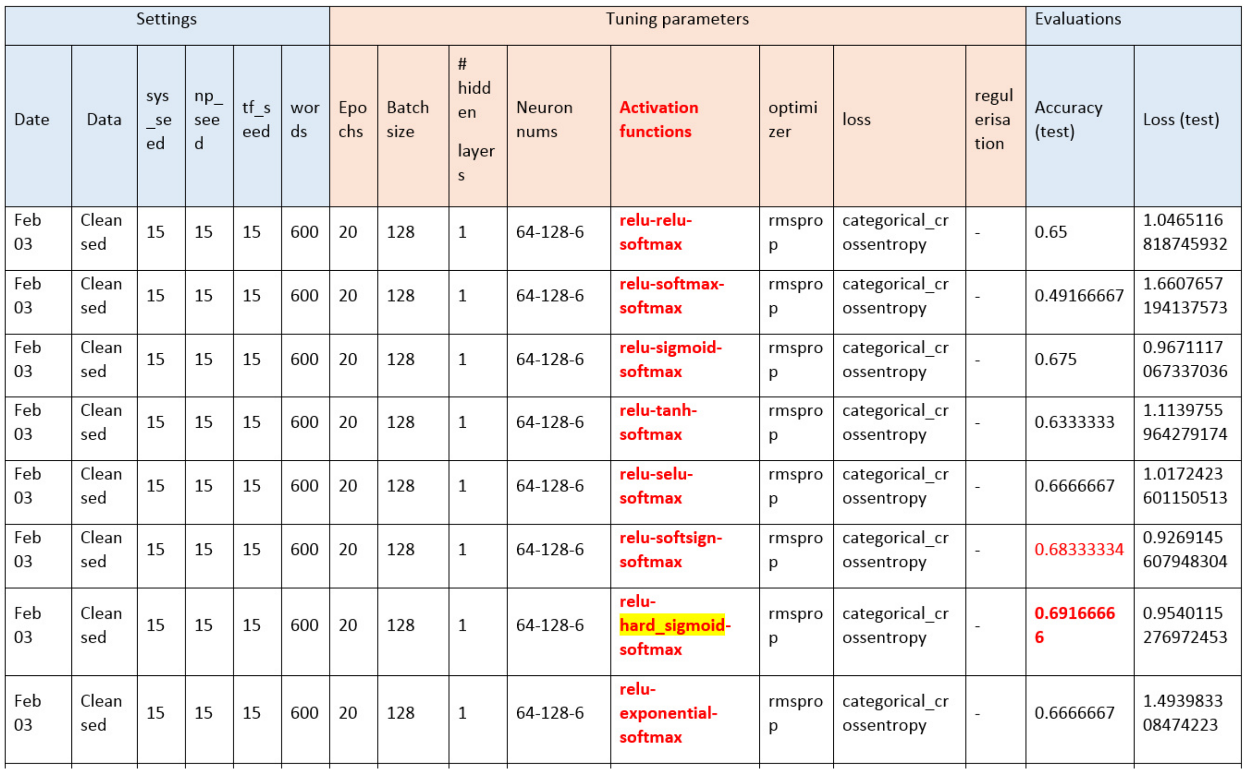
Inputs: activation function list *act_list*, Number of inner layers *num_layers_inner*, *batch_size*, neuron number *neuron*, number of words *words*, *random_state*

```
1:  if length(act_list) does not equal to (num_layers_inner + 2):
2:    print("You should enter two matching lengths. ")
3:  else if length(act_list) < 2:
4:    print("Must have at least 2 activation functions. ")
5:  else:
6:    model ← create a keras Sequential model
      model.add Dense layer(64 neurons, activation←act_list[0], input_shape ← (words,))
      i ← 0
      while i < num_layers_inner:
        model.add Dense layer(neuron neurons, activation←act_list[i+1])
        i ← i+1
      model.add Dense layer(6 neurons, activation←the last element in act_list)
      compile model, with optimizer←'rmsprop', loss←'categorical_crossentropy', metrics←'accuracy'
      split data into training and validation data with ratio 8:2, random state←random_state
      history ← model.fit(training and validation data, batch size←batch_size, epochs←20, )
      results ← model.evaluate(the cleansed X test data, the cleansed Y test data)
      combine ← [results, history]
```

Output: *combine*, which contains both the test loss & accuracy values, and the *history*

The experiment results for activtion function as an example is listed in the Table 4.6-1. We see when there is only one hidden layer, when other hyperparameters are fixed, hard_sigmoid performs the best with the best test accuracy and a relative low loss. Please see my code file for the complete explanation of all the experiments.

Table 4.6-1 Experiments Records - Activation Functions



Settings						Tuning parameters								Evaluations	
Date	Data	sys_seed	np_seed	tf_seed	words	Epochs	Batch size	# hidden layers	Neuron nums	**Activation functions**	optimizer	loss	regularisation	Accuracy (test)	Loss (test)
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-relu-softmax**	rmsprop	categorical_crossentropy	-	0.65	1.0465116818745932
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-softmax-softmax**	rmsprop	categorical_crossentropy	-	0.49166667	1.6607657194137573
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-sigmoid-softmax**	rmsprop	categorical_crossentropy	-	0.675	0.9671117067337036
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-tanh-softmax**	rmsprop	categorical_crossentropy	-	0.6333333	1.1139755964279174
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-selu-softmax**	rmsprop	categorical_crossentropy	-	0.6666667	1.0172423601150513
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-softsign-softmax**	rmsprop	categorical_crossentropy	-	0.68333334	0.9269145607948304
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-hard_sigmoid-softmax**	rmsprop	categorical_crossentropy	-	**0.69166666**	0.9540115276972453
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-exponential-softmax**	rmsprop	categorical_crossentropy	-	0.6666667	1.493983308474223

4.7 Use GridSearchCV to Bruteforce Search

Scikit-learn provides for us with GridSearchCV to conveniently set searching scopes. After setting the searching scope as in Settings 4.7-1, I have got the best hyperparameters: {'act_func': 'tanh', 'batch_size': 256, 'epochs': 20, 'loss_f': 'mean_absolute_error', 'neuron_num1': 64, 'neuron_num2': 128, 'optimizer': 'rmsprop'} with the best accuracy of 0.5521.

Settings 4.7-1: GridSearchCV_01

the input layer neuron number: [32, 64],
the neuron number in each inner layer (identical): [128, 256, 512],
the activation function in each layer (identical): ['relu', 'tanh', 'sigmoid', 'selu'],
batch size: [64, 128, 256, 512],
epoch number: [20, 40, 70],
loss function: ['categorical_crossentropy', 'mean_absolute_error'],
optimizer: ['adam', 'rmsprop', 'sgd', 'adagrad']

Using the hyperparameters shown in Settings 4.7-2, I have got the best settings of: {'act_func': 'relu', 'batch_size': 256, 'epochs': 60, 'loss_f': 'mean_absolute_error', 'neuron_num1': 64, 'neuron_num2': 256, 'num_layers': 0, 'optimizer': 'adam'} with the accuracy of 0.5625.

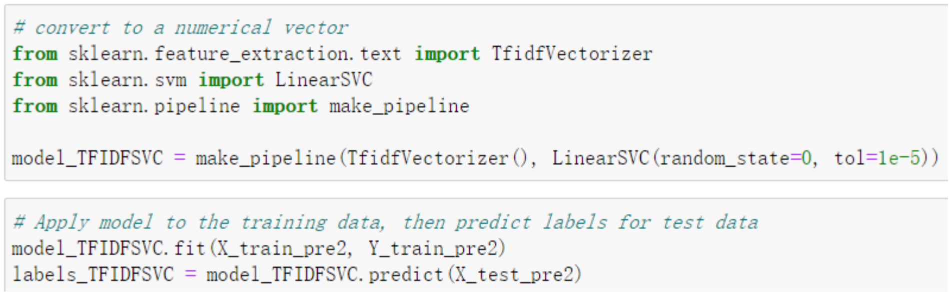
Settings 4.7-2: GridSearchCV_02

the input layer neuron number: [32, 64],
the neuron number in each inner layer (identical): [128, 256, 512],
number of inner layers: [0, 1, 2, 3],
the activation function in each layer (identical): ['relu', 'tanh', 'selu'],
batch size: [128, 256, 512],
epoch number: [20, 40, 60],
loss function: ['categorical_crossentropy', 'mean_absolute_error'],

optimizer: ['adam', 'rmsprop', 'adamax']

4.8 General NLP Models with Vectorizers and Classifiers

Here, I combine various pairs of vectorizers and classifiers to examine each model's performance. For each model, I can import the vectorizer and classifier objects from Scikit-learn, then use make_pipline function to make my model. Then, simply use the 'fit' method for the pipeline model to fit the cleansed training data, and use it to predict the test data, see an example in Fig 4.8-1. I have investigated the following vectorizers: TF-IDF, Count Vectorizer, Bigram; and these classifiers: Support Vector Classifier (SVC), Multinomial Naïve Bayes (MNB), Random Forest, K-Nearest Neighbors; 12 models in total. After building each model, I have evaluated it with heatmap, classification report, ROC curves and AUC.



```python
# convert to a numerical vector
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline

model_TFIDFSVC = make_pipeline(TfidfVectorizer(), LinearSVC(random_state=0, tol=1e-5))
```

```python
# Apply model to the training data, then predict labels for test data
model_TFIDFSVC.fit(X_train_pre2, Y_train_pre2)
labels_TFIDFSVC = model_TFIDFSVC.predict(X_test_pre2)
```

(Fig 4.8-1 Using make_pipeline to Construct a TFIDF-SVC Model)

4.9 1D-Convolutional Neural Network (CNN)

I have used CNN to get rather good results, with accuracy of 0.633 on the test set (Feb 08), fine-tuned a set of settings mentioned by person 'Alialili AIhub' in Bilibili video website [39]. My settings for the best CNN model are shown in Settings 4.9-1. Please see the Fig 5.2-1 in Appendix for its shapes of layers.

Settings 4.9-1: 1D CNN

Layer types: (1) Embedding(2000, 128, 600); (2) Conv1D(256, 3); (3) MaxPooling1D(3, 3);
(4) Conv1D(32, 3, 'relu'); (5) Flatten; (6) Dropout(0.3); (7) BatchNormalization;
(8) Dense(256, 'relu'); (9) Dropout(0.2); (10) Dense(6, 'softmax').
np_seed: 15; tf_seed: 15; words: 600; batch size: 64; epoch: 31

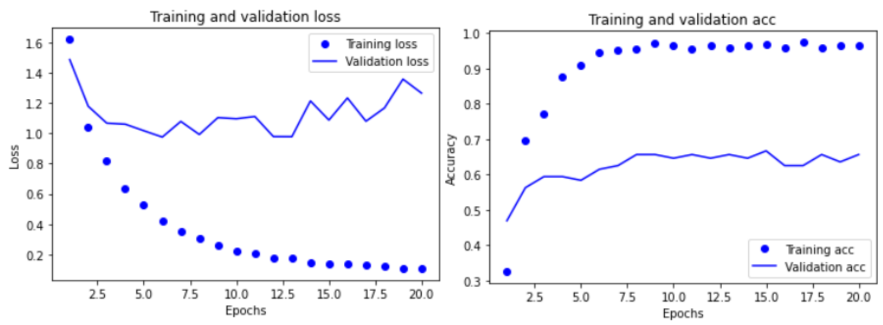
4.10 Recurrent Neural Network (RNN), LSTM Layer

Using Chollet's settings [12] in his book Ch.6.4, I only get a passed accuracy score of 0.267 which is not significantly higher than the baseline 0.167. This model uses two LSTM layers and two Dropout layers.

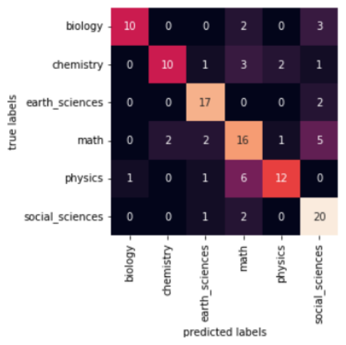
5 Evaluation

5.1 Evaluation of Models

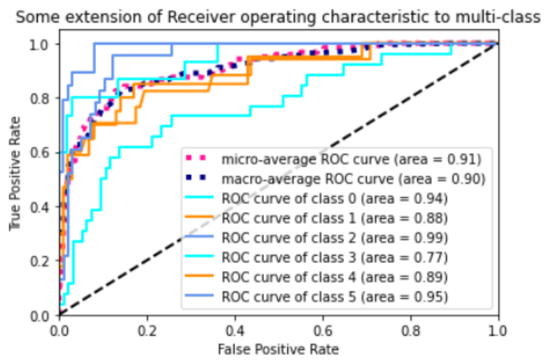
5.1.1 Fully-Connected Neural Networks



(Fig 5.1.1-1 Loss and Accuracy Curves for the Best FCNN Model)



(Fig 5.1.1-2 Heatmap for the Best FCNN)



(Fig 5.1.1-3 ROC for the Best FCNN Model)

In the controlled experiments, the best accuracy score which achieves 0.73 using Hold-Out on the test set, is given by a very simple model when I develop a set of experiemts for the number of layers. It has no hidden layers, with neuron numbers of 256 and 6 in the input and output layers, relu and softmax activation functions for these two layers, rmsprop optimizer, categorical crossentropy loss function, no regularisation, on 20 epochs with input of 600 keywords on the cleansed data, on all random seeds (system, numpy, tensorflow) of 15. From Fig 5.1.1-1, there is no obvious rising-up on the validation loss curve, and no obvious decreasing on the validation accuracy curve, we can say that it is not overfitting yet. From Fig 5.1.1-2 we see the heatmap has distinct colours gathered at the diagnal line; from Fig 5.1.1-3 the ROC curves bended much, and the micro-avg AUC can be 0.91, so it is an excellent model.

All the models with high accuracy on the test set are given in Table 5.1.1-1 (see Appendix). The results for a later repetition for these models and using K-Fold Cross Validation on these models are given in Table 5.1.1-2 (see Appendix). Please see my Jupyter Notebook code file for the complete evaluaion for these models.

Findings for each hyperparameter: **Batch size** – based on my controlled settings, the best performing batch size is 64 when operated on Dec 09, with accuracy=0.675. I have found an increasing-then-decreasing pattern of the accuracies when the batch size goes down. **Number of layers** – the best number of layers is unpredictable, varying from different controlled settings (I use 3 groups), and different operating date (two days). The best layer number covers all from 0 to 4. Therefore, this hyperparameter depends on the real settings of our model. **Activation functions** – from the two groups of experiment, I find that putting hard sigmoid inside the inner layer achieves the best accuracy, and softsign and sigmoid perform the second and third best; if using two same activation functions inside the inner layers, then use tanh. Never try to put softmax inside inner layers, since it has got 0.49 and 0.29 when using once and twice. Others are just viable as well. **Neuron numbers** – in the same settings, two neuron number groups win with 0.68 accuracy: 128-128-256-6; 512-256-128-6. **Optimizers** – Adam stands out with accuracy of 0.708, followed by AdaMax of 0.692. Rmsprop is the third best. Never try to use SGD or AdaGrad! They only have accuracy of 0.425 and 0.358 respectively. AdaDelta is even more unacceptable, with only 0.175 accuracy. **Regularisation** – For L1 and L2 regularizers, if I set the same regularisation for every layer, I can get a high accuracy when L1 or L2 is very small, at around 0.00001 or 0.0001; but L1 cannot be set as big, because l1(0.1) only gets 0.208 accuracy. On the contrary, when l2 is large, l2(0.01) can get the highest accuracy of 0.692. If use l1_l2 combined, we should not set either one as big.

Using GridSearchCV from Scikit-learn, I have found out some of the generally good performing models. The accuracies for the two best models I have found, are 0.55 and 0.56. See Fig 5.1.1-4 in Appendix for their settings.

5.1.2 Classical NLP Models

Surprisingly, using Hold-Out sets, some of the models achieve much higher accuracies than any general neural network model. In Table 5.1.2-1, we can see the highest accuracies can be 0.86, both using SVM Classifier. Multinomial Naïve Bayes Classifier follows up, with an accuracy of under 0.80. K-Nearest Neighbors Classifier with TF-IDF is also brilliant. Bigram with SVM Classifier or MNB performs rather ordinary, with the score similar to the best accuracy score I can achieve using feed-forward neural networks. Never try to use Bigram with K-Nearest Neighbors, because in the heatmap, I find the model classifies almost all the samples as the math category, which is what I cannot explain. Bigram with Random Forest should be avoided as well. Please see my code file for the heatmap, ROC and classification report for each mentioned model.

Using K-Fold Cross Validation (5-fold), though most of the models' accuracies drop down for 2 to 11 percentage points, I get almost the same accuracy rank for these models (see Table 5.1.2-2 and Table 5.1.2-3 in Appendix).

Overall, to choose **classifiers**: Support Vector Classifier is our No.1 choice; MNB Classifier is also within our consideration; if we want to use K-Nearest Neighbors, then we must carefully choose the vectorizer (should be TF-IDF); Random Forest Classifier is only acceptable. To choose **vectorizers**: TF-IDF is our No.1 choice, which performs always very well; Count Vectorizer is almost as good as TF-IDF, but don't combine it with K-Nearest Neighbors Classifier; Bigram performs the worst among these three vectorizers, in all 4 classifiers, perhaps this is why Kandimalla et al. do not use N-grams [18].

Table 5.1.2-1 Performances of My NLP Models Using Hold-out Cross Validation

Date	Random seeds				Vectorizer	Classifier	Cross validation method	Make pipeline ?	Accuracy	Precision (weighted avg)	Recall (weighted avg)	F1 (weighted avg)
	random.seed	np.random.seed	tf.random.set_seed	classifier random_state								
Dec 09	15	15	15	0	TF-IDF	Support Vector Classifier	Hold-out	Yes	0.86	0.87	0.86	0.85
Dec 09	15	15	15	0	Count Vectorizer	Support Vector Classifier	Hold-out	Yes	0.86	0.87	0.86	0.85
Dec 09	15	15	15	0	Bigram	Support Vector Classifier	Hold-out	Yes	0.68	0.70	0.68	0.68
Jan 31	15	15	15	-	TF-IDF	Multinomial Naive Bayes	Hold-out	Yes	0.78	0.80	0.78	0.76
Jan 31	15	15	15	-	Count Vectorizer	Multinomial Naive Bayes	Hold-out	Yes	0.78	0.79	0.78	0.78
Jan 31	15	15	15	-	Bigram	Multinomial Naive Bayes	Hold-out	Yes	0.74	0.75	0.74	0.72
Feb 21	15	15	15	15	TF-IDF	Random Forest	Hold-out	Yes	0.73	0.74	0.72	0.71
Feb 21	15	15	15	15	Count Vectorizer	Random Forest	Hold-out	Yes	0.69	0.68	0.69	0.68
Feb 21	15	15	15	15	Bigram	Random Forest	Hold-out	Yes	0.54	0.64	0.54	0.56
Feb 21	15	15	15	-	TF-IDF	K-Nearest Neighbors	Hold-out	Yes	0.82	0.86	0.82	0.82
Feb 21	15	15	15	-	Count Vectorizer	K-Nearest Neighbors	Hold-out	Yes	0.55	0.60	0.55	0.56
Feb 21	15	15	15	-	Bigram	K-Nearest Neighbors	Hold-out	Yes	0.26	0.54	0.26	0.18

5.1.3 CNN

The highest accuracy I have got so far using CNN, is 0.633 on the test set using hold-out sets, where I have adopted a pre-trained model showed by 'Alialili' in Bilibili video website [39]. This model uses many dropout layers, and one BatchNormalization layer. Because when experimenting, I have not run ROC codes, therefore later I cannot get the ROC curves for the same result since there are randomness within NN models; though, I have AUC=0.85 for this micro-avg ROC when it gets accuracy=0.583, which means this is an outstanding model. Later I have used K-Fold Cross Validation (10-fold) and get an accuracy=0.531 with a 95% confidence interval in between (+/- 4.39%). It is surprising to see a 10 percent drop off by using K-Fold. Apparently, K-Fold is more convincing in that it can utilise the dataset more comprehensively and train-and-validate many times. Although this model cannot beat the feature engineering models at all, it is quite a progress where I can significantly win the baseline 0.167.

Compared with my initial CNN model which only gets accuracy of 0.125, Alialili's model [39] uses Batch Normalization, some Dropout layers, and the proper size of input and output size in each MaxPooling and Conv1D layer, and thus his model can get a better result. Dropout and Batch Normalization aim to prevent overfitting and have improved the performance of test set.

5.1.4 RNN and LSTM

These models I have developed perform not as well as the CNN models. Generally, I get an accuracy score of around 0.267, not exceeding too much from the baseline 0.167.

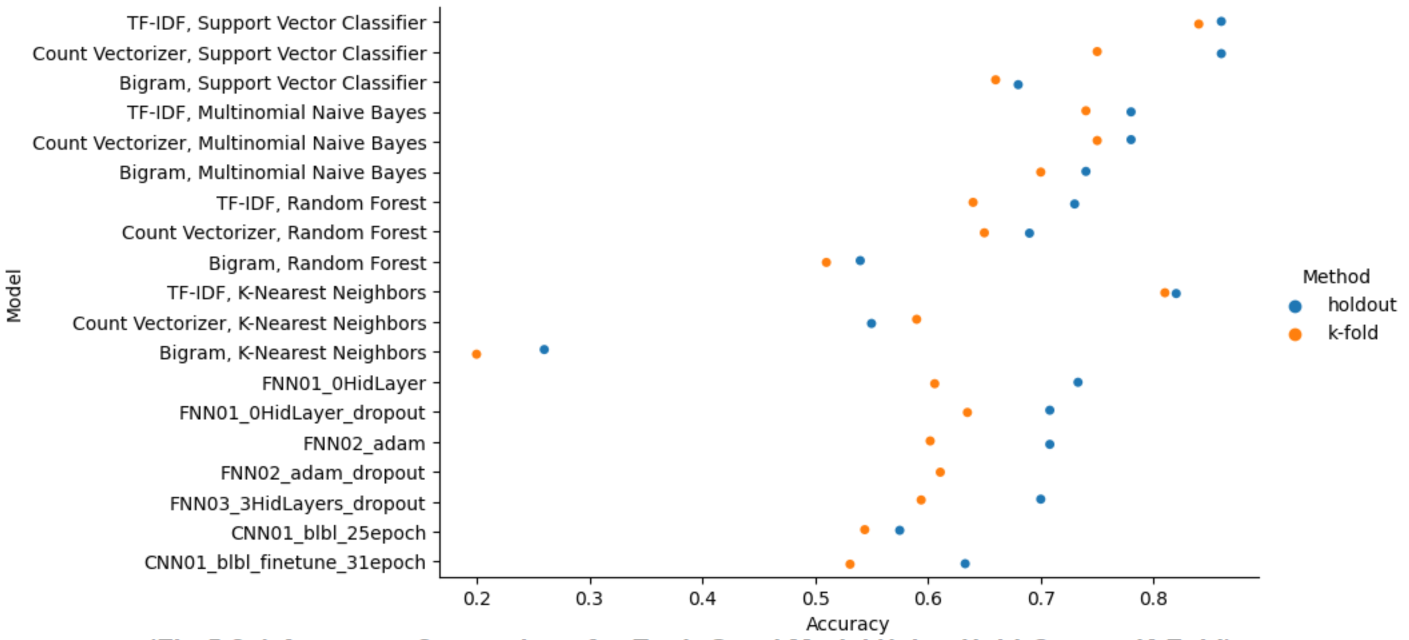
I have added an LSTM layer to an RNN, fine-tuning the settings given by Chollet [12] in his book Ch.6; however his model does not perform well in my case. Using the fine-tuned AWD-LSTM settings given by Merity in his Github, I only get accuracy=0.167 on the test set (Feb 04), which is just the baseline.

5.1.5 Latent Dirchlet Allocation

In the Fig 5.1.5-1 (see Appendix), our LDA has classified the whole dataset based on a completely different mechanism from the scientific-subject-based rubric which we use, perhaps because LDA takes into account all of the words, and it has combined both the subject classification and research method classification together. The result groups generated on Oct 25 seem like: Topic 0 – general administrative issue; Topic 1 – Cell Biology; Topic 2 – Physics; Topic 3 – Cell Biology experiment; Topic 4 – Geography; Topic 5 – Physics experiment.

5.2 Comparison of Models

Among the 19 models listed in Fig 5.2-1, 18 models are acceptable as a pre-trained model for later researches, because all of them have an accuracy of above 0.50, which is much higher than the commonsense baseline 0.167. The exception is the model using 'Bigram Vectorizer with KNN Classifier' which is only a little bit better than the commonsense baseline.



(Fig 5.2-1 Accuracy Comparison for Each Good Model Using Hold-Out vs. K-Fold)

The best Neural Network models cannot compare with other NLP models. The best Neural Network models in general can achieve an accuracy of 0.733 using Hold-Out (and 0.635 using K-Fold), while the best NLP models can get 0.86 using Hold-Out (and 0.84 using K-Fold). Among the Neural Networks, the best Fully-Connected Neural

Networks win against the best CNN models. Among the Neural Networks, using Dropout layers can have a slightly better accuracy than not using them.

5.3 The Best Model(s)

From Fig 5.2-1, we see the best model is the 'TF-IDF Vectorizer with SVC Classifier' model, with both the Hold-Out accuracy and K-Fold accuracy almost 0.9 (about 0.85). There are two second-best models: the 'Count Vectorizer with SVC Classifier' model, and 'TF-IDF Vectorizer with KNN Classifier' model. The former has a very high accuracy using Hold-Out (0.86), but has an approximately 10-percentage-points lower accuracy using K-Fold (0.75). The latter has very close accuracies using both Hold-Out and K-Fold, which are about 0.82. There are two third-best models: the 'TF-IDF Vectorizer with MNB Classifier' model, and 'Count Vectorizer with MNB Classifier' model, with accuracy of above 0.75 for both their Hold-Out score and K-Fold score. See Table 5.1.2-1, Table 5.1.2-2, and Table 5.1.2-3 for numerical details. The best FCNN is the one which achieves accuracy=0.733 using hold-out, with Settings 5.3-1; using K-Fold Cross Validation (10-fold), it has accuracy=0.606 with 95% confidence interval in between (+/-4.79%); if with one Dropout(0.3) layer added, it can achieve accuracy=0.635 using K-Fold.

Settings 5.3-1: The Best Fully-Connected Neural Network

All layers are Dense. 2 layers in total. Neuron numbers: 256-6; random seeds 15;
activation function: relu-softmax; optimizer: rmsprop; batch size=64; epochs=20;
loss func: categorical_crossentropy. Operated on Feb 03.

The best Deep Neural Network I find, is a CNN model, with Settings 5.3-2. It achieves the accuracy=0.633 using Hold-Out, 0.531 using K-Fold; and when the epoch number=25 and using K-Fold (10-fold), it achieves accuracy=0.544 with 95% confidence interval within (+/-3.54%). See the Fig 5.3-1 in Appendix for its shapes.

Settings 5.3-2: The Best CNN

There are 10 layers in total. batch size=64; epochs=31; random seeds 15. Operated on Feb 08.

```
1:  layers.Embedding(input_dim=2000, output_dim=128, input_length=600)
2:  layers.Conv1D(256, 3, padding='same', activation='relu')
3:  layers.MaxPooling1D(3, 3, padding='same')
4:  layers.Conv1D(32, 3, padding='same', activation='relu')
5:  layers.Flatten()
6:  layers.Dropout(0.3)
7:  layers.BatchNormalization()
8:  layers.Dense(256, activation='relu')
9:  layers.Dropout(0.2)
10: layers.Dense(6, activation='softmax') optimizer='rmsprop', loss='categorical_crossentropy',
   loss='categorical_crossentropy', metrics=['accuracy']
```

5.4 Evaluation of This Project

5.4.1 Successes

The collected database is balanced. Have controlled random seeds, which helps other people to repeat my steps, and it is a controlled variable. I have adapted codes to draw the ROC curves for each category and Micro- and Macro-avg, can show the AUC for each. Have splitted the training set and test set which have consistent data (the number of features – only the article body; and the numerical value distribution of each feature), and are complete in each category. Have implemented K-Fold Cross Validation in both NLP models and NN models, which can compensate for my relatively small sample size. Have presented all-correct data cleansing procedures, including using the correct order of expanding contractions, lemmatisation, etc. This is a crucial step to eliminate noises that may lead to model overfitting by misguiding it to ignore the core. Have compared the performances of many FCNN after hyperparameter tuning, many DNN including CNN, RNN and others, and general NLP models. Have tried to use pre-trained models or settings given by other people. Have attempted to repeat the original codes given by prestigious papers, including ULMFiT (FastAI), AWD-LSTM, Wikipedia2Vec, etc.

5.4.2 Limitations and Future Extensions

I can repeat the NLP models, but cannot repeat the same result using the same Neural Network models; it is perhaps because of the randomness of embedding layer, model weight, and Dropout layer, etc. It would have been better to use K-Fold in all of my controlled experiments, though it would be slow; better to find the codes for ROC and K-Fold earlier, and then to use these evaluation methods at a time, since NN models are unrepeatable. I have tried to use NNI [28] to search for the best hyperparameters on another small model, but no codes now for my models. FastAI or ULMFiT: Although I have tried to run FastAI pre-training on their datasets, I find their algorithm runs extremely slowly on two of my devices; after learning the data, the calling to use the learning data stuck. AWD-LSTM: From their github code repository [20], I stuck when running codes and got the CUDA error; after deleting all the .cuda() and .cpu() codes, there still are errors. Many of the pre-trained models require a specific format of data input, for which I can hardly find an example in them; there are UTF-8 decoding errors. It would be better if I can adopt an NLP-combined layer inside my neural network, since human-taught lessons are more targeted; furthermore, I hope to tell my machine 'just to classify based on subjects, not on experiment methods'. I should extend my database in order to include all kinds of academic journal articles, to let my data to be more representative. Should connect the article data with different journal platforms, different publishing websites, and other metadata which I have collected using the EventRegistry API, to get more interesting discoveries.

6 Conclusion

6.1 Summary of This Project

During these months, I have explored a classification dataset in the topic of 'academic journal articles' which is collected by me, by using different Machine Learning methods and models, with comprehensive evaluation metrics, implemented experiments, and finally filtered out the best models and have made rather comprehensive discussions. Since few literature reviews can be found in the 'academic journal articles' topic, my findings can be seen as a set-up foundation for all the later researches in this topic. 18 of the 19 models shown in Fig 5.2-1 has surpassed the baseline 0.167 in a great amount, and thus they can be used as pre-trained models; once with some fine-tunings, these models can be used in related researches.

Although great effort has been put into the hyperparameter-adjusting stage in my feed-forward neural network, till now, general NLP models outperform Feed-Forward Neural Networks. The best accuracy I can get by tuning hyperparameters, is approximately 0.733 using Hold-Out sets, which has an accuracy of 0.606 (+/- 4.79%) using K-Fold (can be higher using a Dropout layer); while NLP methods with the vectoriser which either puts attention to the importance of words (TF-IDF) or uses one-hot encoding (Count Vectorizer), and the classifier either considers previous knowledge (Naïve Bayes) or draws lines among samples in hyperspaces (SVC), all perform quite impressive – they all can get an accuracy score of over 0.74 or even 0.86 (the exception is Bigram vectorizer), just using the built-in pipeline of sklearn. For Deep Neural Networks, it is better to have a pre-trained settings for either CNN or RNN and then fine-tune some of the hyperparameters and values. I have gained benefits from previously existing models, since my best DNN model is a CNN with accuracy=0.633 using Hold-Out (and accuracy=0.531 using K-Fold) for which I have adopted a pre-trained model used by a video in Bilibili.

Some detailed conclusions/discoveries: using Dropout layers can increase the accuracy of K-Fold Cross Validation; many models tend to mis-classify other articles as 'math', and it is surprising to see most articles in 'Bigram KNN' model have been mis-classified as 'math'; almost all the models perform somewhat worse using K-Fold Cross Validation than using Hold-Out; in a NN model, the complexity of layers is not a decisive factor for a high accuracy; Bigram is the most unssuccessful vectorizer; it is interesting why KNN varies much by using different vectorizers (is it because of 'lazy learning'?); …More discoveries can be found in my code file.

Though I have not discovered why Neural Networks cannot beat some of the NLP models, I guess it is because neural networks are without language-targeted teaching from humans, and thus the tuning is completely technical and rather blind; while NLP models are written by language experts so they have taught the machine how to learn in a better way, and the machine has a rough direction. This postulation should be examined by further efforts.

Practically, for the whole academic journal platforms, editors can adopt my models to classify their articles based on subjects, but I suggest them to add a step to manually review, since my models cannot guarantee a 90% accuracy, and we don't want any mis-classified paper to be put into wrong sub-journals.

6.2 Discussions

What about multi-class classification? Consider an example: a biology paper containing experimental words and some descriptions about the research society. Should it be classified as 'Research Society' or 'Biological Experiment' or just general 'Biology'? Should it depend on the ratio of the contents? How many labels should we give each article – can the number vary from one to another?

It would be more comprehensive if other languages were considered. The differences can be in word-detection, encoding type, (without) lemmatization, etc. for example in character-based Chinese language.

'The NFL (No Free Lunch Theorem) stated that within certain constraints, over the space of all possible problems, every optimization technique will perform as well as every other one on average (including Random Search)'[37]. The implication is that we must discuss the model performances within a specific learning task; it supports that my project can be seen as a foundation for this 'journal articles classification' task.

In the aspect of article classification, we can classify them based on subjects, and we can also classify them into theoretical, experimental, descriptive, or designed papers, based on research methods (though they are not applicable for social sciences). We may have layers of classifications for natural sciences. In my model, I may further purify the feature words used in my NLP models or purify the cleansed data in order to eliminate the impact of research-methods-based classification towards my subject-based classification.

Having a broader view, we may have some doubts for the entire Machine Learning realm. NN can answer 'What' questions, but cannot answer 'How' and 'Why' questions. Even feature engineering cannot answer 'Why' questions. Thus, Machine Learning apparently does not have the 'super-evolutionary speedup' as mentioned by Pearl and Mackenzie [10]. ML algorithms also will fall into Berkson's Paradox [10]: ML have a tendency to make misleading causation conclusions between two unrelated objects based solely on a strong link.

Sawyer highlighted the moral implications of AI in his BBC programme [33]. At an interview for a well-known cosmetics brand, an AI program weeds out some of the top prospects. As a result, I should ban anybody from using my models to undertake a weeding process for academic publications, given there are individuals researching the weeding process [15].

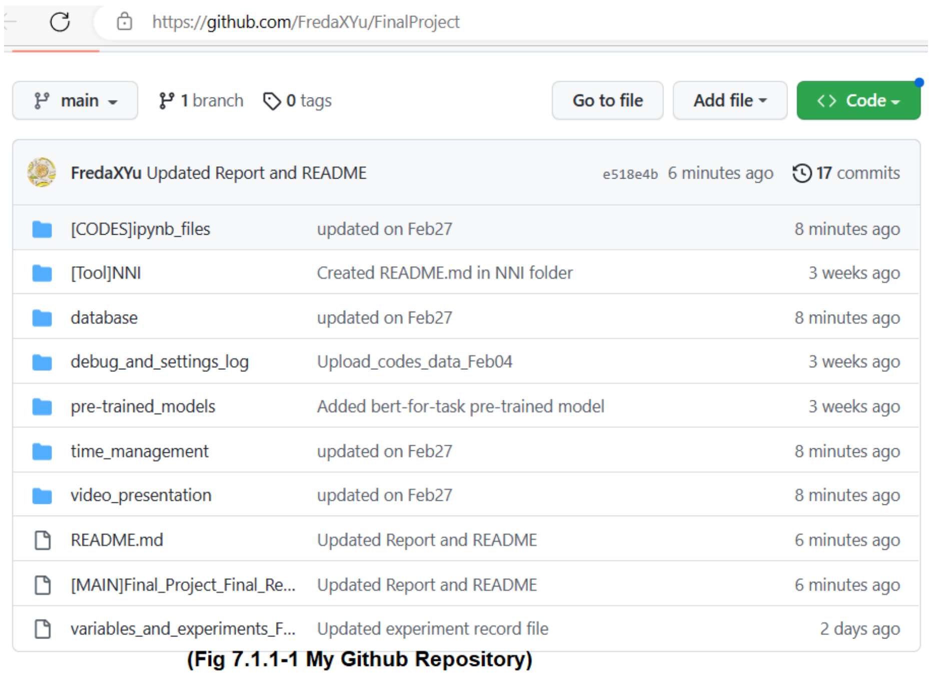
Is classification really a crucial task in our life? Cramer, in his book *Chaos and Order* Chapter 1, has stated that 'Mother Nature does not create categories' but she 'creates individuals', and we humans are 'shortsighted to seek for similarities' so that we may 'notice each item'[6]. Therefore, the 'classification' idea might have been exaggerated at all when it was born in Darwin's era. We must be aware – the classification is just a convenient way for us to read information; it is not the goal itself. For article classification, the the safest case is that we have human editors to filter for categories, and to ensure that each unique article would not be buried. For instance in my project, almost all the models more or less tend to mis-classify other articles as 'math' subject, which is not allowed in the journal peer review process. Therefore, AI surely can develop deeply in classification field using evolving algorithms, but the wise work of human is what can never be totally replaced.

7 Appendix

7.1 Source Codes & Files Repository

7.1.1 Repository

- Github: https://github.com/FredaXYu/FinalProject (see Fig 7.1.1-1 for the repository, and Fig 7.1.1-2 for the Git log)



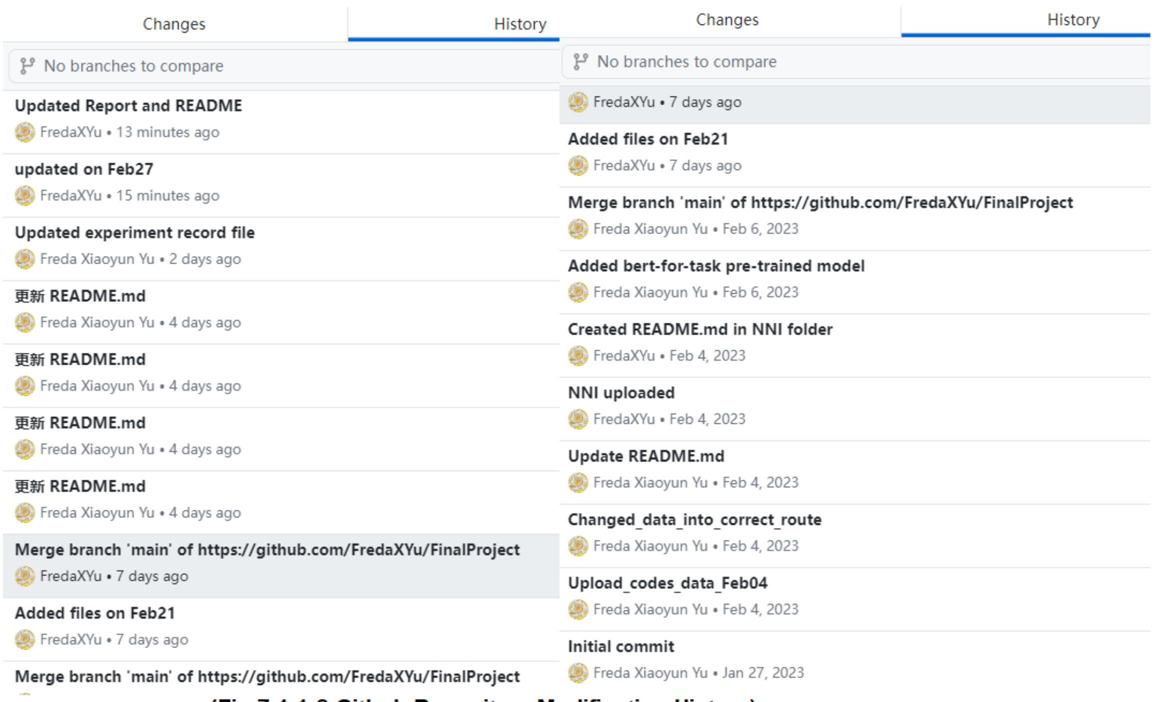
https://github.com/FredaXYu/FinalProject

main · 1 branch · 0 tags · Go to file · Add file · Code

FredaXYu Updated Report and README · e518e4b 6 minutes ago · 17 commits

[CODES]ipynb_files	updated on Feb27	8 minutes ago
[Tool]NNI	Created README.md in NNI folder	3 weeks ago
database	updated on Feb27	8 minutes ago
debug_and_settings_log	Upload_codes_data_Feb04	3 weeks ago
pre-trained_models	Added bert-for-task pre-trained model	3 weeks ago
time_management	updated on Feb27	8 minutes ago
video_presentation	updated on Feb27	8 minutes ago
README.md	Updated Report and README	6 minutes ago
[MAIN]Final_Project_Final_Re...	Updated Report and README	6 minutes ago
variables_and_experiments_F...	Updated experiment record file	2 days ago

(Fig 7.1.1-1 My Github Repository)



Changes · History

No branches to compare

```
Updated Report and README
FredaXYu • 13 minutes ago
updated on Feb27
FredaXYu • 15 minutes ago
Updated experiment record file
Freda Xiaoyun Yu • 2 days ago
更新 README.md
Freda Xiaoyun Yu • 4 days ago
更新 README.md
Freda Xiaoyun Yu • 4 days ago
更新 README.md
Freda Xiaoyun Yu • 4 days ago
更新 README.md
Freda Xiaoyun Yu • 4 days ago
Merge branch 'main' of https://github.com/FredaXYu/FinalProject
FredaXYu • 7 days ago
Added files on Feb21
FredaXYu • 7 days ago
Merge branch 'main' of https://github.com/FredaXYu/FinalProject
```

Changes · History

No branches to compare

```
FredaXYu • 7 days ago
Added files on Feb21
FredaXYu • 7 days ago
Merge branch 'main' of https://github.com/FredaXYu/FinalProject
Freda Xiaoyun Yu • Feb 6, 2023
Added bert-for-task pre-trained model
Freda Xiaoyun Yu • Feb 6, 2023
Created README.md in NNI folder
FredaXYu • Feb 4, 2023
NNI uploaded
FredaXYu • Feb 4, 2023
Update README.md
Freda Xiaoyun Yu • Feb 4, 2023
Changed_data_into_correct_route
Freda Xiaoyun Yu • Feb 4, 2023
Upload_codes_data_Feb04
Freda Xiaoyun Yu • Feb 4, 2023
Initial commit
Freda Xiaoyun Yu • Jan 27, 2023
```

(Fig 7.1.1-2 Github Repository Modification History)

- (Alternative) Google Drive:

https://drive.google.com/drive/folders/1v36uUfWtJI9Grjm-tAGryEQ6y2t2B-De?usp=share_link

7.1.2 Sources on Sections

- Data: https://github.com/FredaXYu/FinalProject/tree/main/database including Collected data and Reformed data.

- Codes: https://github.com/FredaXYu/FinalProject/tree/main/%5BCODES%5Dipynb_files (**the main code file is named like '[MAIN]Final_Project_Scientific_Articles.ipynb'**)

- Files:

Experiment records: https://github.com/FredaXYu/FinalProject/tree/main/%5BIMPORTANT%5Dexperiment_records

Timetable: https://drive.google.com/file/d/1BreFfxayTG5MiKJAptcTda2ogAQvkBIz/view?usp=sharing (Alternative: https://github.com/FredaXYu/FinalProject/tree/main/time_management)

- Pitch Video

Video link: 【Final_Project_Video_Demonstration_FredaXiaoyunYu】 https://www.bilibili.com/video/BV1CY4y127YB/?share_source=copy_web&vd_source=296c14837e03501f00801a512d70f87e

7.2 References

[1] Buschman, John, and Gloria J. Leckie. 2009. *Information Technology in Librarianship: New Critical Approaches*. Libraries Unlimited Incorporated.

[2] Cope, Bill, and Angus Phillips. 2014. *The Future of the Academic Journal*. Chandos Publishing.

[3] Dewey, Melvil. 1876. Preface. *A Classification and Subject Index, for Cataloguing and Arranging the Books and Pamphlets of a Library*. Brick Row Book Shop, Project Gutenberg (4 June 2004).

[4] Gross, Alan G., et al. 2011. *Communicating Science: The Scientific Article from the 17th Century to the Present*. Parlor Press.

[5] Svensson, Goran. 2006. *Academic Journals and Academic Publishing*. Emerald Group Publishing.

[6] Cramer, Friedrich. 1993. *Chaos and Order: The Complex Structure of Living Systems*. VCH.

[7] Abrizah, A., et al. 13 July 2012. *LIS Journals Scientific Impact and Subject Categorization: A Comparison between Web of Science and Scopus*. Scientometrics, vol. 94, no. 2, pp. 721–740. https://doi.org/10.1007/s11192-012-0813-7.

[8] Boyack, Kevin W., and Richard Klavans. 2022. *Accurately Identifying Topics Using Text: Mapping PubMed1*. STI 2018 Conference Proceedings, pp. 107–115. https://hdl.handle.net/1887/65319, Accessed 7 Nov. 2022.

[9] Cortes-Rodriguez, Patricio, et al. 10 Sept. 2021. *Methodology to Categorize the Research Output of Academic Institutions, through the Sustainable Development Goals (Sdgs)*. Emerald Open Research, vol. 3, pp. 21. https://doi.org/10.35241/emeraldopenres.14355.1.

[10] Dahlberg, Ingetraut. 12 Sept. 1976. *Classification Theory, Yesterday and Today*. International Classification, vol. 3, no. 2, pp. 85–90. https://doi.org/10.5771/0943-7444-1976-2-85.

[11] Pearl, Judea, and Dana Mackenzie. 2020. *The Book of Why: The New Science of Cause and Effect*. Basic Books.

[12] Chollet, Francois. 2021. *Deep Learning with Python*. O'REILLY.

[13] Howard, Jeremy, and Sebastian Ruder. 2018. *Universal Language Model Fine-Tuning for Text Classification.* Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). https://doi.org/10.18653/v1/p18-1031.

[14] Parrochia, Daniel. *Classification*. Internet Encyclopedia of Philosophy (IEP). https://iep.utm.edu/classification-in-science/.

[15] Wagstaff, Kiri L., and Geoffrey Z. Liu. 2018. *Automated Classification to Improve the Efficiency of Weeding Library Collections.* The Journal of Academic Librarianship, vol. 44, no. 2, pp. 238–247. https://doi.org/10.1016/j.acalib.2018.02.001.

[16] Unknown. *Introduction to Library Classification*. National Institute of Open Schooling. https://nios.ac.in/media/documents/vocational/CLS/Certificate_Course_in_Library_Science_english/M2_PDF/M2L2.pdf.

[17] Ambalavanan, Ashwin Karthik, and Murthy V. Devarakonda. 2020. *Using the Contextual Language Model Bert for Multi-Criteria Classification of Scientific Articles*. Journal of Biomedical Informatics, vol. 112, p. 103578. https://doi.org/10.1016/j.jbi.2020.103578.

[18] Kandimalla, Bharath, et al. 10 Feb. 2021. *Large Scale Subject Category Classification of Scholarly Papers with Deep Attentive Neural Networks*. Frontiers in Research Metrics and Analytics, vol. 5. https://doi.org/10.3389/frma.2020.600382.

[19] Lee, Kangwook, et al. 4 Dec. 2017. *A Discourse-Aware Neural Network-Based Text Model for Document-Level Text Classification*. Journal of Information Science, vol. 44, no. 6, pp. 715–735. https://doi.org/10.1177/0165551517743644.

[20] Merity, Stephen, Nitish Shirish Keskar, and Richard Socher. 2017. *Regularizing and optimizing LSTM language models.* arXiv preprint, arXiv:1708.02182(2017). https://github.com/salesforce/awd-lstm-lm.

[21] Feng, Xiang, et al. 16 Mar. 2020. *Academic Emotion Classification and Recognition Method for Large-Scale Online Learning Environment—Based on A-Cnn and LSTM-Att Deep Learning Pipeline Method.* International Journal of Environmental Research and Public Health, vol. 17, no. 6, p. 1941. https://doi.org/10.3390/ijerph17061941.

[22] Garcia-Gorrostieta, Jesús Miguel, and Aurelio López-López. 24 May 2018. *Argument Component Classification in Academic Writings*. Journal of Intelligent & Fuzzy Systems, vol. 34, no. 5, pp. 3037–3047. https://doi.org/10.3233/jifs-169488.

[23] Glänzel, W., et al. 1999. *An Item-by-Item Subject Classification of Papers Published in Journals Covered by the SSCI Database Using Reference Analysis*. Scientometrics, vol. 46, no. 3, pp. 431–441. https://doi.org/10.1007/bf02459602. https://akjournals.com/view/journals/11192/46/3/article-p431.xml.

[24] Gómez-Núñez, Antonio J., et al. 14 Aug. 2011. *Improving SCImago Journal & Country Rank (SJR) Subject Classification through Reference Analysis*. Scientometrics, vol. 89, no. 3, pp. 741–758. https://doi.org/10.1007/s11192-011-0485-8 .

[25] Milojević, Staša. 2020. *Practical Method to Reclassify Web of Science Articles into Unique Subject Categories and Broad Disciplines*. Quantitative Science Studies, vol. 1, no. 1, pp. 183–206. https://doi.org/10.1162/qss_a_00014.

[26] Osborne, Francesco, et al. 2016. *Automatic Classification of Springer Nature Proceedings with Smart Topic Miner*. Lecture Notes in Computer Science, International Semantic Web Conference, pp. 383–399. https://doi.org/10.1007/978-3-319-46547-0_33. https://link.springer.com/chapter/10.1007/978-3-319-46547-0_33.

[27] Zins, Chaim. 2007. *Classification Schemes of Information Science: Twenty-Eight Scholars Map the Field*. Journal of the American Society for Information Science and Technology, vol. 58, no. 5, pp. 645–672. https://doi.org/10.1002/asi.20506.

[28] Microsoft. Neural Network Intelligence (version v2.10). https://github.com/microsoft/nni.

[29] Rie Johnson and Tong Zhang. 2015. *Effective use of word order for text categorization with convolutional neural networks*. In NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 103–112.

[30] Zhou, Chunting, et al. 30 Nov. 2015. *A C-LSTM Neural Network for Text Classification*. International Journal of Emerging Trends in Engineering Research, vol. 2. https://arxiv.org/abs/1511.08630.

[31] jcblaisecruz02. 6 Dec. 2020. AWD-LSTM: AWD-LSTM and Ulmfit Reproduction from Scratch. GitHub - JCBLAISECRUZ02. https://github.com/jcblaisecruz02/awd-lstm.

[32] Olah, Christopher. 27 Aug. 2015. Understanding LSTM Networks. Colah's Blog, Github. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[33] Sawyer, Georgia. 5 June 2022. *Computer Says No*. Performance by Daniel Henry, BBC Three, BBC. https://www.bbc.co.uk/programmes/m0015gvw, Accessed 19 Nov. 2022.

[34] Lunqing Hou, Fei Wang, Xin Deng, and Zhouan Shi. 2020. *Tensorflow 从零开始学 (Tensorflow Learning From Zero)*. Publishing House of Electronics Industry, Beijing.

[35] DataTechNotes. 2020. Classification Example with Linear SVC in Python. https://www.datatechnotes.com/2020/07/classification-example-with-linearsvm-in-python.html.

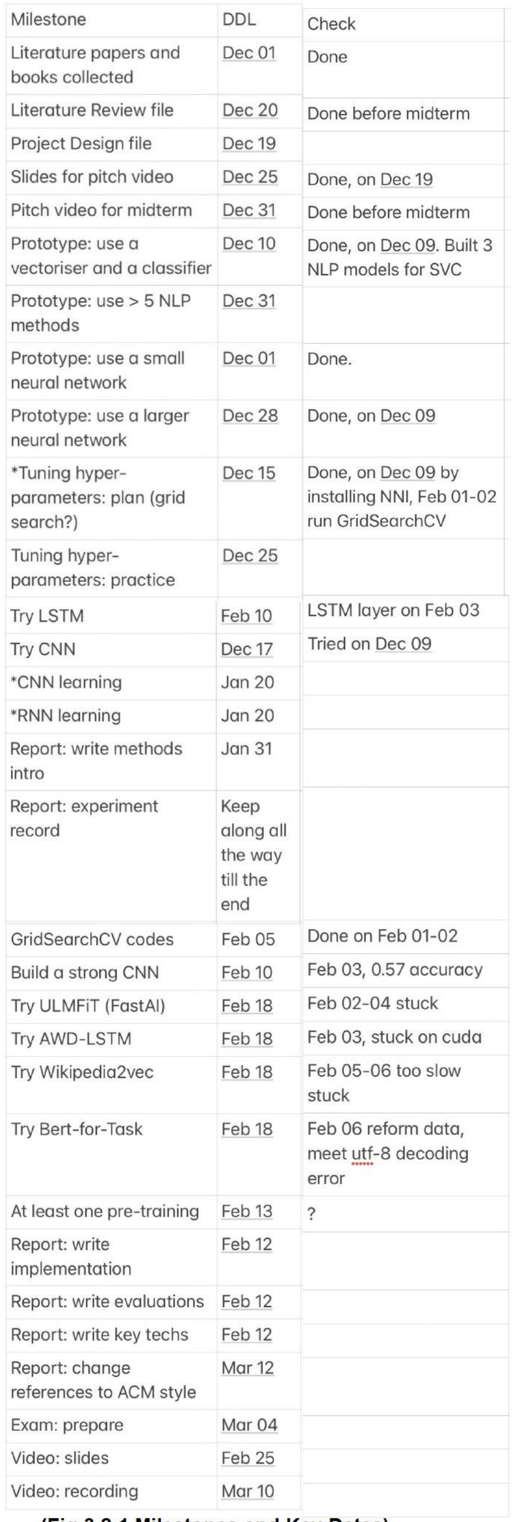
[36] PublicKnowdgeProject. Unknown Year. Issue and Article Metadata - Better Practices in Journal Metadata. https://docs.pkp.sfu.ca/metadata-practices/en/issue-article-metadata.

[37] Luke, Sean. Feb. 2016. *Essentials of Metaheuristics*. Lulu, Online Version 2.3, pp. 221. https://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf.

[38] jiangxinyang227. 2019. Applications of BERT and ALBERT in Downstream Tasks (BERT 和 ALBERT 在下游任务中的应用). BERT for Task. https://github.com/jiangxinyang227/bert-for-task.

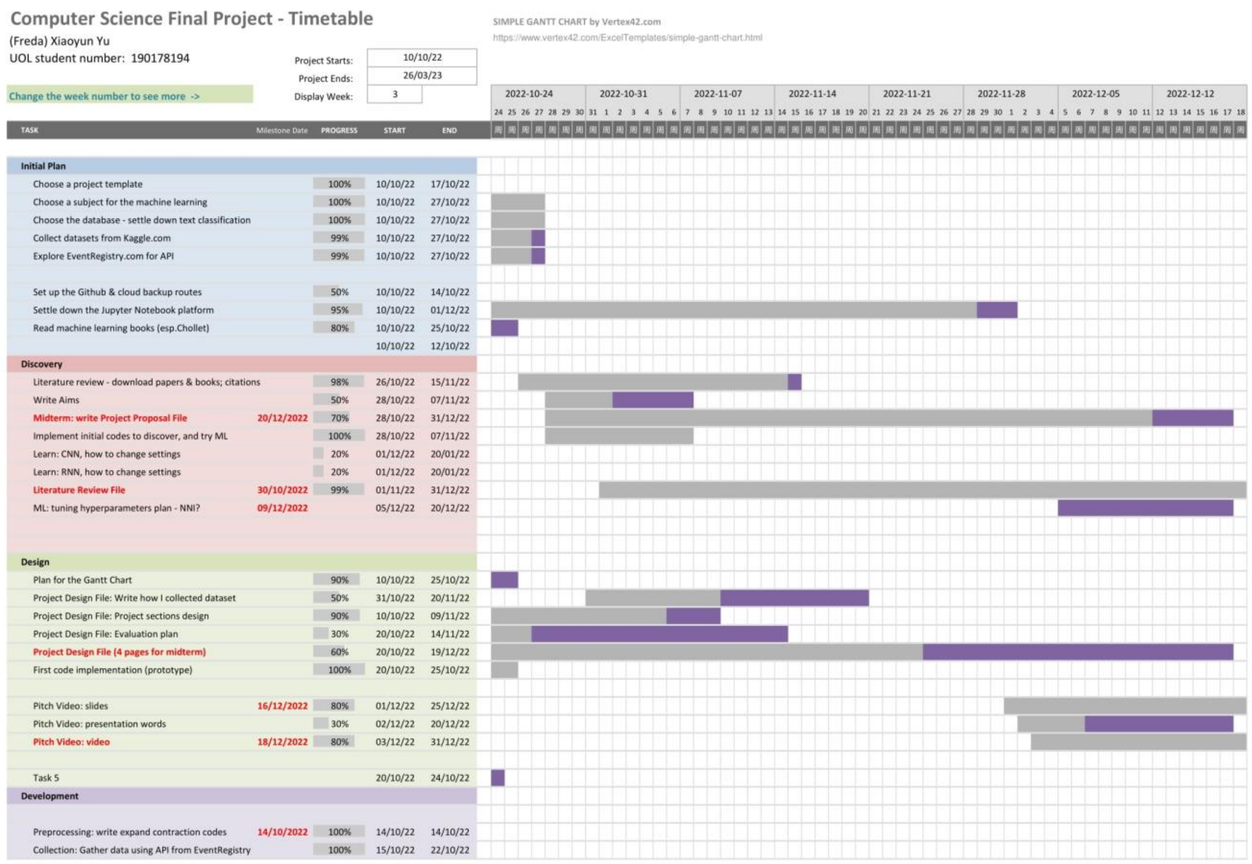
[39] Alialili (阿力阿哩哩). 19 Mar 2020. CNN Convolutional Neural Network and Text Classification (CNN 卷积神经网络与文本分类(keras)). Video. Retrieved from Bilibili, https://www.bilibili.com/video/BV1u7411d7zU/?share_source=copy_web&vd_source=296c14837e03501f00801a512d70f87e.

7.3 Additional Tables & Pictures



Milestone	DDL	Check
Literature papers and books collected	Dec 01	Done
Literature Review file	Dec 20	Done before midterm
Project Design file	Dec 19	
Slides for pitch video	Dec 25	Done, on Dec 19
Pitch video for midterm	Dec 31	Done before midterm
Prototype: use a vectoriser and a classifier	Dec 10	Done, on Dec 09. Built 3 NLP models for SVC
Prototype: use > 5 NLP methods	Dec 31	
Prototype: use a small neural network	Dec 01	Done.
Prototype: use a larger neural network	Dec 28	Done, on Dec 09
*Tuning hyper-parameters: plan (grid search?)	Dec 15	Done, on Dec 09 by installing NNI, Feb 01-02 run GridSearchCV
Tuning hyper-parameters: practice	Dec 25	
Try LSTM	Feb 10	LSTM layer on Feb 03
Try CNN	Dec 17	Tried on Dec 09
*CNN learning	Jan 20	
*RNN learning	Jan 20	
Report: write methods intro	Jan 31	
Report: experiment record	Keep along all the way till the end	
GridSearchCV codes	Feb 05	Done on Feb 01-02
Build a strong CNN	Feb 10	Feb 03, 0.57 accuracy
Try ULMFiT (FastAI)	Feb 18	Feb 02-04 stuck
Try AWD-LSTM	Feb 18	Feb 03, stuck on cuda
Try Wikipedia2vec	Feb 18	Feb 05-06 too slow stuck
Try Bert-for-Task	Feb 18	Feb 06 reform data, meet utf-8 decoding error
At least one pre-training	Feb 13	?
Report: write implementation	Feb 12	
Report: write evaluations	Feb 12	
Report: write key techs	Feb 12	
Report: change references to ACM style	Mar 12	
Exam: prepare	Mar 04	
Video: slides	Feb 25	
Video: recording	Mar 10	

(Fig 3.2-1 Milestones and Key Dates)



Computer Science Final Project - Timetable

(Freda) Xiaoyun Yu
UOL student number: 190178194

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Starts: 10/10/22
Project Ends: 26/03/23
Display Week: 3

Change the week number to see more ->

TASK	Milestone Date	PROGRESS	START	END
Initial Plan				
Choose a project template		100%	10/10/22	17/10/22
Choose a subject for the machine learning		100%	10/10/22	27/10/22
Choose the database - settle down text classification		100%	10/10/22	27/10/22
Collect datasets from Kaggle.com		99%	10/10/22	27/10/22
Explore EventRegistry.com for API		99%	10/10/22	27/10/22
Set up the Github & cloud backup routes		50%	10/10/22	14/10/22
Settle down the Jupyter Notebook platform		95%	10/10/22	01/12/22
Read machine learning books (esp.Chollet)		80%	10/10/22	25/10/22
			10/10/22	12/10/22
Discovery				
Literature review - download papers & books; citations		98%	26/10/22	15/11/22
Write Aims		50%	28/10/22	07/11/22
Midterm: write Project Proposal File	20/12/2022	70%	28/10/22	31/12/22
Implement initial codes to discover, and try ML		100%	28/10/22	07/11/22
Learn: CNN, how to change settings		20%	01/12/22	20/01/22
Learn: RNN, how to change settings		20%	01/12/22	20/01/22
Literature Review File	30/10/2022	99%	01/11/22	31/12/22
ML: tuning hyperparameters plan - NNI?	09/12/2022		05/12/22	20/12/22
Design				
Plan for the Gantt Chart		90%	10/10/22	25/10/22
Project Design File: Write how I collected dataset		50%	31/10/22	20/11/22
Project Design File: Project sections design		90%	10/10/22	09/11/22
Project Design File: Evaluation plan		30%	20/10/22	14/11/22
Project Design File (4 pages for midterm)		60%	20/10/22	19/12/22
First code implementation (prototype)		100%	20/10/22	25/10/22
Pitch Video: slides	16/12/2022	80%	01/12/22	25/12/22
Pitch Video: presentation words		30%	02/12/22	20/12/22
Pitch Video: video	18/12/2022	80%	03/12/22	31/12/22
Task 5			20/10/22	24/10/22
Development				
Preprocessing: write expand contraction codes	14/10/2022	100%	14/10/22	14/10/22
Collection: Gather data using API from EventRegistry		100%	15/10/22	22/10/22

(Fig 3.2-2 Timetable Snapshot)

```
num_words_input = 600 # We should set the max number of crucial words to be identified.
tok = keras.preprocessing.text.Tokenizer(num_words=num_words_input, lower=True, char_level=False)
tok.fit_on_texts(X_train_pre) # fit tokenizer to our training text data
X_train = tok.texts_to_matrix(X_train_pre)
X_test = tok.texts_to_matrix(X_test_pre)
```

```
encoder = LabelEncoder()
encoder.fit(Y_train_pre)
Y_train = encoder.transform(Y_train_pre)
Y_test = encoder.transform(Y_test_pre)
# Convert the labels to a one-hot representation:
num_classes = np.max(Y_train) + 1
Y_train = keras.utils.to_categorical(Y_train, num_classes)
Y_test = keras.utils.to_categorical(Y_test, num_classes)
```

```
Y_train[300]  # check. It should be a one-hot encoding array
```

```
array([0., 0., 0., 1., 0., 0.], dtype=float32)
```

(Fig 4.3-1 Encoding, Feed to Keras)

```
Best hyperparameters are: {'act_func': 'tanh', 'batch_size': 256, 'epochs': 20, 'loss_f': 'mean_absolut
e_error', 'neuron_num1': 64, 'neuron_num2': 128, 'optimizer': 'rmsprop'} Best score is:  0.552083333333
3333
Best hyperparameters are: {'act_func': 'relu', 'batch_size': 256, 'epochs': 60, 'loss_f': 'mean_absolut
e_error', 'neuron_num1': 64, 'neuron_num2': 256, 'num_layers': 0, 'optimizer': 'adam'} Best score is:
0.5625
```

(Fig 5.1.1-4 GridSearchCV Results)

Table 5.1.1-1 Summary of High Accuracy Feed-Forward Neural Network Models

(In each row, the cell with red bold characters is the independent variable.)

Settings						Tuning parameters								Evaluations	
Date	Data	sys_seed	np_seed	tf_seed	words	Epochs	Batch size	# hidden layers	Neuron nums	Activation functions	optimizer	loss	regularisation	Accuracy (test)	Loss (test)
Feb 03	Cleansed	15	15	15	600	20	**64**	**0**	**256-6**	relu-softmax	rmsprop	categorical_crossentropy	-	0.7333333492279053	0.9777682423591614
Feb 03	Cleansed	15	15	15	600	20	128	2	512-256-128-6	relu-tanh-tanh-softmax	**adam**	categorical_crossentropy	-	0.7083333	1.2403667132059732
Jan 31	Cleansed	15	15	15	600	20	**256**	**3**	**128-128-128-128-6**	relu-relu- relu-relu-softmax	rmsprop	categorical_crossentropy	-	0.699999988079071	1.0144932270050049
Jan 31	Cleansed	15	15	15	600	20	**128**	**0**	**128-6**	relu-softmax	rmsprop	categorical_crossentropy	-	0.699999988079071	0.9842715859413147
Jan 31	Cleansed	15	15	15	600	20	**128**	**2**	**128-128-128-6**	relu- relu-relu-softmax	rmsprop	categorical_crossentropy	-	0.699999988079071	0.9746896624565125
Jan 31	Cleansed	15	15	15	600	20	**64**	**1**	**256-256-6**	relu-relu-softmax	rmsprop	categorical_crossentropy	-	0.6916666626930237	1.1270121335983276
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-hard_sigmoid-softmax**	rmsprop	categorical_crossentropy	-	0.69166666	0.9540115276972453
Feb 03	Cleansed	15	15	15	600	20	128	2	512-256-128-6	relu-tanh-tanh-softmax	**adamax**	categorical_crossentropy	-	0.69166666	1.0454886635144551
Jan 31	Cleansed	15	15	15	600	20	**256**	**1**	**128-128-6**	relu-relu-softmax	rmsprop	categorical_crossentropy	-	0.6833333373069763	0.9424988031387329
Feb 03	Cleansed	15	15	15	600	20	**64**	**1**	**256-256-6**	relu-relu-softmax	rmsprop	categorical_crossentropy	-	0.6833333373069763	1.1629196405410767
Feb 03	Cleansed	15	15	15	600	20	**64**	**2**	**256-256-256-6**	relu- relu-relu-softmax	rmsprop	categorical_crossentropy	-	0.6833333373069763	1.229711651802063
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-softsign-softmax**	rmsprop	categorical_crossentropy	-	0.68333334	0.9269145607948304
Feb 03	Cleansed	15	15	15	600	15	512	2	**128-128-256-6**	relu-tanh-tanh-softmax	rmsprop	categorical_crossentropy	-	0.68333334	0.9293365836143493
Feb 03	Cleansed	15	15	15	600	15	512	2	**512-256-128-6**	relu-tanh-tanh-softmax	rmsprop	categorical_crossentropy	-	0.68333334	0.9334465821584066
Dec 09	Cleansed	15	15	15	600	20	**64**	1	64-64-6	relu-relu-softmax	rmsprop	categorical_crossentropy	-	0.6750	1.1273
Jan 31	Cleansed	15	15	15	600	20	**64**	**4**	**256-256-256-256-256-6**	relu-relu-relu-relu- relu -softmax	rmsprop	categorical_crossentropy	-	0.675000011920929	1.114376425743103
Jan 31	Cleansed	15	15	15	600	20	**64**	**5**	**256-256-256-256-256-256-6**	relu- relu-relu-relu-relu- relu -softmax	rmsprop	categorical_crossentropy	-	0.675000011920929	1.5121153593063354
Jan 31	Cleansed	15	15	15	600	20	**256**	**0**	**128-6**	relu-softmax	rmsprop	categorical_crossentropy	-	0.675000011920929	0.9862571954727173
Feb 03	Cleansed	15	15	15	600	20	**128**	**0**	**128-6**	relu-softmax	rmsprop	categorical_crossentropy	-	0.675000011920929	0.9399003386497498
Feb 03	Cleansed	15	15	15	600	20	128	1	64-128-6	**relu-sigmoid-softmax**	rmsprop	categorical_crossentropy	-	0.675	0.9671117067337036

Table 5.1.1-2 Repetition for the High-Performing FNN Models Using K-Fold Cross Validation

Settings						Tuning parameters								Evaluations	
Date	Data	sys_seed	np_seed	tf_seed	words	Epochs	Batch size	# hid layers	Neuron nums	Activation functions	optimizer	loss	regulerisation	Accuracy Holdout (test)	Avg Accuracy 10-Fold (val)
Feb 28	Cleansed	15	15	15	600	20	**64**	**0**	**256-6**	relu-softmax	rmsprop	categorical_crossentropy	-	0.7083333134651184	60.63% (+/- 4.79%)
Feb 23	Cleansed	15	15	15	600	20	**64**	**0**	**256-6**	relu-softmax	rmsprop	categorical_crossentropy	Dropout(0.3)	0.7083333134651184	63.54% (+/- 6.67%)
Feb 28	Cleansed	15	15	15	600	20	128	2	512-256-128-6	relu-tanh-tanh-softmax	**adam**	categorical_crossentropy	-	0.6833333373069763	60.21% (+/- 5.70%)
Feb 28	Cleansed	15	15	15	600	20	128	2	512-256-128-6	relu-tanh-tanh-softmax	**adam**	categorical_crossentropy	2 Dropout	-	61.04% (+/- 6.85%)
Feb 28	Cleansed	15	15	15	600	20	**256**	**3**	**128-128-128-128-6**	relu-relu-relu-relu-softmax	rmsprop	categorical_crossentropy	3 Dropout	0.699999988079071	59.38% (+/- 6.86%)

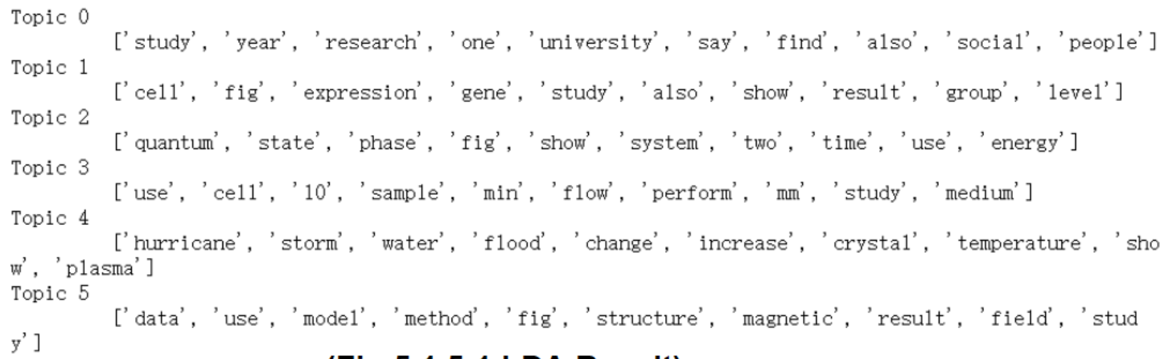
Table 5.1.2-2 Performances of My NLP Models Using K-Fold Cross Validation

Date	Random seeds				Vectorizer	Classifier	Cross validation method	Make pipeline?	Error?	Avg accuracy with 95% confidence interval
	random.seed	np.random.seed	tf.random.set_seed	classifier random_state						
Feb 28	15	15	15	0	TF-IDF	Support Vector Classifier	5-fold	yes	-	0.84 (+/- 0.05)
Feb 28	15	15	15	0	Count Vectorizer	Support Vector Classifier	5-fold	yes	warning	0.75 (+/- 0.09)
Feb 28	15	15	15	0	Bigram	Support Vector Classifier	5-fold	yes	warning	0.66 (+/- 0.05)
Feb 28	15	15	15	-	TF-IDF	Multinomial Naive Bayes	5-fold	yes	-	0.74 (+/- 0.08)
Feb 28	15	15	15	-	Count Vectorizer	Multinomial Naive Bayes	5-fold	yes	-	0.75 (+/- 0.08)
Feb 28	15	15	15	-	Bigram	Multinomial Naive Bayes	5-fold	yes	-	0.70 (+/- 0.03)
Feb 28	15	15	15	15	TF-IDF	Random Forest	5-fold	yes	-	0.64 (+/- 0.02)
Feb 28	15	15	15	15	Count Vectorizer	Random Forest	5-fold	yes	-	0.65 (+/- 0.04)
Feb 28	15	15	15	15	Bigram	Random Forest	5-fold	yes	-	0.51 (+/- 0.08)
Feb 28	15	15	15	-	TF-IDF	K-Nearest Neighbors	5-fold	yes	-	0.81 (+/- 0.07)
Feb 28	15	15	15	-	Count Vectorizer	K-Nearest Neighbors	5-fold	yes	-	0.59 (+/- 0.14)
Feb 28	15	15	15	-	Bigram	K-Nearest Neighbors	5-fold	yes	-	0.20 (+/- 0.03)

Table 5.1.2-3 Performances of My NLP Models Using Hold-Out Cross Validation vs. K-Fold Cross Validation

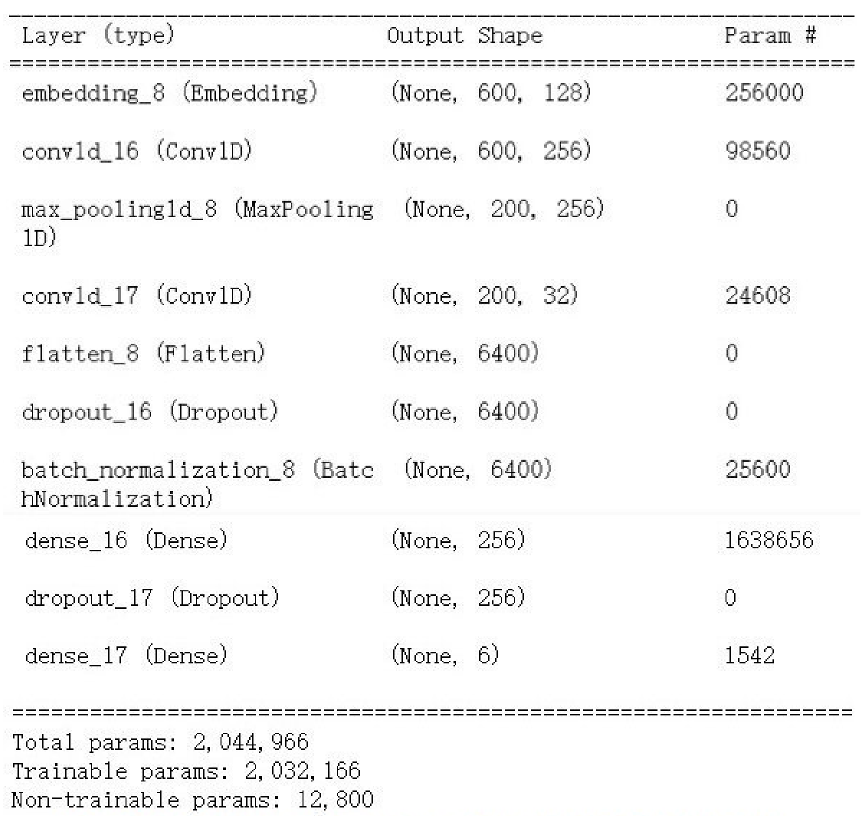
Vectorizer	Classifier	Avg Accuracy using Hold-out	Avg Accuracy using 5-Fold
TF-IDF	Support Vector Classifier	0.86	0.84 (+/- 0.05)
Count Vectorizer	Support Vector Classifier	0.86	0.75 (+/- 0.09)
Bigram	Support Vector Classifier	0.68	0.66 (+/- 0.05)
TF-IDF	Multinomial Naive Bayes	0.78	0.74 (+/- 0.08)
Count Vectorizer	Multinomial Naive Bayes	0.78	0.75 (+/- 0.08)
Bigram	Multinomial Naive Bayes	0.74	0.70 (+/- 0.03)
TF-IDF	Random Forest	0.73	0.64 (+/- 0.02)
Count Vectorizer	Random Forest	0.69	0.65 (+/- 0.04)
Bigram	Random Forest	0.54	0.51 (+/- 0.08)
TF-IDF	K-Nearest Neighbors	0.82	0.81 (+/- 0.07)
Count Vectorizer	K-Nearest Neighbors	0.55	0.59 (+/- 0.14)
Bigram	K-Nearest Neighbors	0.26	0.20 (+/- 0.03)

(Oct 25)



```
Topic 0
        ['study', 'year', 'research', 'one', 'university', 'say', 'find', 'also', 'social', 'people']
Topic 1
        ['cell', 'fig', 'expression', 'gene', 'study', 'also', 'show', 'result', 'group', 'level']
Topic 2
        ['quantum', 'state', 'phase', 'fig', 'show', 'system', 'two', 'time', 'use', 'energy']
Topic 3
        ['use', 'cell', '10', 'sample', 'min', 'flow', 'perform', 'mm', 'study', 'medium']
Topic 4
        ['hurricane', 'storm', 'water', 'flood', 'change', 'increase', 'crystal', 'temperature', 'sho
w', 'plasma']
Topic 5
        ['data', 'use', 'model', 'method', 'fig', 'structure', 'magnetic', 'result', 'field', 'stud
y']
```

(Fig 5.1.5-1 LDA Result)



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_8 (Embedding)     (None, 600, 128)          256000

 conv1d_16 (Conv1D)          (None, 600, 256)          98560

 max_pooling1d_8 (MaxPooling  (None, 200, 256)         0
 1D)

 conv1d_17 (Conv1D)          (None, 200, 32)           24608

 flatten_8 (Flatten)         (None, 6400)              0

 dropout_16 (Dropout)        (None, 6400)              0

 batch_normalization_8 (Batc  (None, 6400)             25600
 hNormalization)
 dense_16 (Dense)            (None, 256)               1638656

 dropout_17 (Dropout)        (None, 256)               0

 dense_17 (Dense)            (None, 6)                 1542

=================================================================
Total params: 2,044,966
Trainable params: 2,032,166
Non-trainable params: 12,800
```

(Fig 5.3-1 Shape of Layers of the Best CNN Model)