

Master Thesis

Automated Prediction of Experimental Metadata from Scientific Publications

Stuti Nayak

Master Thesis DKE-19-06

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science of Data Science for Decision Making
at the Department of Data Science and Knowledge Engineering
of the Maastricht University

Thesis Committee:

Dr. Rachel Cavill
Dr. Jerry Spanakis
Dr. Amrapali Zaveri (IDS)
Dr. Michel Dumontier (IDS)

Maastricht University
Faculty of Science and Engineering
Department of Data Science and Knowledge Engineering

March 1, 2019

Abstract

While there exists an abundance of open biomedical data, the lack of high quality metadata makes it challenging for others to find relevant datasets and to reuse them for another purpose. In particular, metadata are useful to understand the nature and provenance of the data. A common approach to improving the quality of metadata relies on expensive human curation, which itself is time consuming and also prone to error. Towards improving the quality of metadata we use scientific publications to automatically predict metadata key: value. For prediction, we use a Convolutional Neural Network (CNN) and a Bidirectional Long-short term memory network (BiLSTM). Additionally, we also perform a comparison with multi-label classification methods. The best results we get are from CNN. We focus our attention on the NCBI Disease Corpus.

Acknowledgements

First of all, I would like to thank my supervisor Dr. Amrapali Zaveri for introducing me to the problem of metadata quality, her guidance, constant encouragement and support, and invaluable help. Additionally, I would also like to thank Dr. Michel Dumontier for giving me the opportunity to work at IDS and his guidance. I would also like to thank Pedro Hernandez Serrano for helping me out with my doubts throughout the project. I would like to thank Dr. Rachel Cavill and Dr. Jerry Spanakis my supervisors at DKE for supervising me and providing valuable insights. I would also like to thank my friends Gaurav Maheshwari, Priyansh Trivedi and Yash Patel for their helpful comments. Last but not the least, I would like to thank my family who always provided me with perpetual love and support.

Contents

1	Introduction	6
1.1	Context and Motivation	6
1.2	Research Questions and Outline	8
1.2.0.0.1	Contributions	8
1.2.0.0.2	Outline	9
2	Related Work	10
2.1	Biomedical metadata quality assessment	10
2.2	Identifying biomedical terms in text	11
2.2.0.0.1	Topic Modeling	11
2.2.0.0.2	Natural Language Processing (NLP)	12
2.2.0.0.3	Named Entity Recognition (NER) and Deep Learning	12
2.2.0.0.4	Machine Learning	13
3	Preliminaries and Dataset	14
3.1	Preliminaries	14
3.1.1	RDF, Ontology, and SPARQL	14
3.2	Dataset	15
3.2.0.0.1	Data Preprocessing	18
3.2.0.0.2	Text Preprocessing	19
4	Methods for Text Classification	20
4.1	Introduction	20
4.2	Multi-label Classification	21
4.2.1	Problem Transformations	21
4.2.1.0.1	Binary Relevance (BR)	21
4.2.1.0.2	Label Powerset (LP)	22
4.2.1.0.3	Classifier Chains (CC)	22
4.2.2	Problem adaptation and Classifiers	23
4.2.2.0.1	Multi-label k Nearest Neighbours (MLkNN)	23
4.2.2.0.2	Decision Trees (DT)	23
4.2.2.0.3	Random Forest	24
4.2.2.0.4	Multi-layer perceptron (MLP)	24

4.3	Deep Learning	25
4.3.1	Unfolding the architecture	26
4.3.1.0.1	One-hot Encoding	26
4.3.1.0.2	Word Embeddings	26
4.3.1.0.3	Encoder	27
4.3.1.0.4	Neural Networks	27
4.3.1.1	CNN	30
4.3.1.1.1	CNN Model Description	33
4.3.1.2	RNN	35
4.3.1.2.1	RNN Model Description	35
5	Results and Discussion	39
	Performance measures	39
5.1	A baseline approach	41
5.2	Predicting Disease Terms	42
5.2.1	Multi-label Classification	42
5.2.2	Neural Network Methods	42
5.3	Predicting Super Classes	44
5.3.1	Multi-label Classification	46
5.3.2	Neural Network Methods	46
5.4	Discussion	48
6	Conclusions and Future Work	52
6.1	Conclusions	52
6.1.1	Research Question 1 & 2	53
6.1.2	Research Question 3 & 4	53
6.1.3	Research Question 5	54
6.2	Limitations and Future Work	54

List of Figures

3.1	An example of GEO Sample	15
3.2	Example of a bad metadata	16
3.3	Example of a satisfactory metadata	17
3.4	Annotation Process (Source: https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/DISEASE/)	18
4.1	Multi-label problem transformation - Binary Relevance	22
4.2	Multi-label problem transformation - Label Powerset	22
4.3	Multi-label problem transformation - Classifier Chains	23
4.4	Typical neural network based deep learning architecture for multiclass classification of texts.	25
4.5	The figure illustrates how to obtain embedding of a particular word “cat” from a word embedding matrix. The one-hot encoding vector of the word “cat” is multiplied to the word embedding matrix to obtain the embedding vector for <i>cat</i>	27
4.6	This figure illustrates the property of word embeddings where linear vector differences between male-female vector is preserved, that is, constant. That is, $\mathcal{W}(\text{“woman”}) - \mathcal{W}(\text{“man”}) \simeq \mathcal{W}(\text{“aunt”}) - \mathcal{W}(\text{“uncle”}) \simeq \mathcal{W}(\text{“queen”}) - \mathcal{W}(\text{“king”})$ (Image Source [37]). . .	27
4.7	Typical neuron in a neural network, which can be intuitively thought of a building block of the underlying neural network. . .	28
4.8	Vector inputs in the neuron	28
4.9	A simple neural network architecture for the task of multiclass classification, where the output layer is followed by a softmax layer, that takes outputs from the output layer and converts a distribution of numbers into probabilities.	29
4.10	A forward and backward propagation in the neural network. . .	30
4.11	The figure illustrates the components of a typical convolutional neural network (CNN). The figure in the left views CNN as a small number of complex layers, where within each layer a number of operations are applied. While on the right, CNN is viewed as a larger number of simple layers performing elementary operations. . .	31
4.12	This figure illustrates the operation of max-pooling to summarize the feature maps.	31

4.13	This figure illustrates the operation of convolution on a 2D image. Boxes and arrows are drawn which indicate how the convolution takes place when the kernel/filter is applied to the upper-left region of the input.	32
4.14	This figure illustrates a CNN with one convolution layer that is precisely used in the experimental setup in this thesis in the context of biomedical abstracts.	33
4.15	A recurrent neural network with outputs \hat{y} is depicted in the figure. The recurrent neural network processes the information/signal from the input x by incorporating the signal from unit h that is passed on forward in each time step. The figure on the left shows that block diagram of an RNN where the black square denotes a delay of a single time step. While on the right hand side, the block diagram is further unfolded and hence can be seen that each node/unit is now associated with one particular time instance.	35
4.16	The detailed internals of a LSTM unit	37
5.1	Mapping superclasses in MeSH	45

List of Tables

3.1	NCBI disease Corpus characteristics	18
3.2	Input corpus statistics	19
5.1	Confusion Matrix - table that is used to calculate evaluation metrics	39
5.2	An example of predictions in a multi-label classification to depict evaluation metrics	40
5.3	Results for Disease terms using TF-IDF feature selection	42
5.4	Results for Disease terms using Bag of words feature selection	42
5.5	Results for deep neural networks for predicting disease names	43
5.6	Best performing methods when predicting disease terms	43
5.7	Results for predicting disease terms using deep neural nets when only some part of abstract is taken as input	44
5.8	Results of predicting super class using TF-IDF feature selection	46
5.9	Results of predicting super class using Bag of words feature selection	46
5.10	Results for deep neural networks for predicting super class	47
5.11	Best performance among the different methods when predicting super classes	47
5.12	Results for predicting super classes using deep neural nets when only some part of abstract is taken as input	47
5.13	Prediction by CNN for the three abstracts	50
5.14	Prediction by BiLSTM for the three abstracts	50

Chapter 1

Introduction

1.1 Context and Motivation

Enormous amounts of biomedical data have been and are being produced at an unprecedented rate by researchers all over the world [27]. This is mainly due to advancements in molecular technologies that have enabled extensive profiling of biological samples and have unleashed a myriad of so-called ‘*omics*’ data such as gene expression, microRNA expression, DNA methylation, and DNA mutation data. During the last decade or so journals, investigators, funding agencies have realized that this data should be stored, shared with and used by other investigators. However, to enable reuse there is an urgent need to understand the structure of datasets and the experimental conditions under which they were produced [5]. That is, there is an urgent need for accurate, structured and complete description of the data – defined as *metadata*.

While there exists an abundance of open biomedical data, the lack of high quality metadata makes it challenging for others to find relevant datasets and to reuse them for another purpose [18, 28]. This, in turn, can facilitate a data-driven approach by combining and analyzing similar data to uncover novel insights or even more subtle trends in the data. These insights can then be formed into a hypothesis that can be tested in the laboratory [1]. In particular, metadata are useful to understand the nature and provenance of the data. A common approach to improving the quality of metadata relies on expensive human curation, which itself is time consuming and also prone to error. Poor metadata leads to the problems of (i) interpretability - can we understand what was done in the biomedical experiment? and (ii) improved precision and recall in database queries - to find all studies that meet constraints (e.g. all studies for a particular disease) and (iii) re-usability - to use this data for discovery, validation, and reproducibility.

The recently published FAIR principles specify desirable criteria that metadata and their corresponding datasets should meet to be Findable, Accessible, Interoperable, and Reusable [57]. For data to be FAIR, metadata needs to

be accurate and uniform (relying on controlled terms where possible). However, currently there is a large amount of biomedical metadata, which is of poor quality, that is, it is extremely heterogeneous and which makes data re-use extremely difficult [18]. Although there has been a lot of work done in meta-analysis of public omics data. One particular example demonstrated by Khatri and colleagues [31], depicts the power of integrating data from publicly available sources where investigators had used different assay technologies, resulting in the identification of a common mechanism of transplant rejection using only publicly available gene-expression data. They then showed that the gene signatures associated with this mechanism improved diagnosis, and ultimately led to the identification of two drugs previously approved by the FDA that can be repurposed to treat transplant rejection. For Khatri and colleagues, a key impediment to their success was the low quality of the metadata associated with gene expression datasets. Their work required them to tediously examine and curate each and every dataset and associated publication to create metadata pertaining to the organism, tissue, protocol, and other essential parameters, so that they could use datasets that specifically met their inclusion criteria.

One of the major challenge towards assessing and improving the quality of biomedical metadata is the size of data that is present. Delving further into the problem at hand – let’s take an example of Gene Expression Omnibus (GEO) dataset [16] which is a widely used database for cross-species gene expression data. Currently users can submit data to GEO via three ways: (i) Spreadsheets, (ii) SOFT format (plain text), (iii) MINiML format¹ (XML). When users submit data to GEO via a spreadsheet, it requires them to fill out a metadata template that follows the guidelines set out by the Minimum Information About a Microarray Experiment (MIAME) guidelines [6]. GEO allows users to specify metadata in the form of textual key: value pairs (e.g. *sex: female*). However, since there is no structured vocabulary or format available, the 44,000,000+ *key: value* pairs suffer from numerous quality issues such as:

- minor spelling discrepancies
(e.g. `age at diagnosis (years)`, `age at diagonosis (years)`; `genotype/varat`, `genotype/varaiation`, `genotype/variaion` `genotype/variataion`)
- having different syntactic representations (e.g. `age (years)`, `age(yrs)` and `age_year`)
- using different terms altogether to denote one concept (e.g. `disease` vs. `illness` vs. `condition`)
- using two different key terms in one (e.g. `disease/cell type`, `tissue/cell line`, `treatment age`).

Looking at these issues it can be seen that there is an urgent need to solve this research problem which would in-turn facilitate the re-usability of data.

¹‘MIAME Notation in Markup Language’ format

1.2 Research Questions and Outline

Using domain experts for the curation of assessing the quality of metadata is not only time consuming, but also not scalable. Moreover, without a standardized set of terms with which to fill out the template fields (in the form when filling out the metadata), there are different versions of the same term without any (semantic) links between them, thus leading to several quality problems. Thus, there is a need for efficient methods for curating the metadata. In our previous projects [39, 40] we tried to cluster similar terms together using topic modeling and machine learning respectively. But when dealing with the values it is very complex to map them to a concept because of a large amount of data. Whereas in scientific publication, we can identify values and then map them to a respective concept, this can be done per document. Therefore, we could exploit the information present in the scientific publication - where our aim is to automatically predict metadata from scientific publications (unstructured text) using Machine Learning or Deep Learning in other words we want to build a **Metadata Wizard**.

We hypothesize that the scientific publications have in-depth descriptions of the experiments performed which facilitate predicting better quality metadata. The research questions outlined for this project are listed below:

- RQ1: To what extent can we **automatically** predict metadata (key-value pairs) describing experiments from scientific publications?
- RQ2: Which specific metadata keys can be accurately predicted?
- RQ3: What set of features contribute to the accurate prediction of metadata?
- RQ4: Which is the best automated (machine learning/deep learning) method to predict metadata?
- RQ5: To what extent does the accuracy improves when we use abstract and full text?

1.2.0.0.1 Contributions The contributions in this thesis project are to develop a Wizard which tells the metadata when given an unstructured text:

- developed a model for identifying metadata using scientific publications focusing on the metadata category *disease*,
- performed empirical experiments on NCBI disease corpus [14] to identify disease categories and specific disease using different neural network approach and
- performed in-depth analysis of results.

1.2.0.0.2 Outline The thesis is structured as follows, Chapter 1 introduces the problem statement, motivation and research question that the project aims to answer. Chapter 2 reports about the related work that has been done in this area of research and a brief mention of how this thesis project is different with all that work that has already been done mentioning the innovation of this approach. Chapter 3 describes the method that has been performed including explanations about the terminologies used. In Chapter 4 we describe the model specification in the context of Natural language processing (NLP). Chapter 5 discusses the results and the analysis of those results. Lastly, chapter 6 discusses a brief summary, conclusions drawn, the limitations and the future work of the project.

Chapter 2

Related Work

The sections in this chapter are divided into two, namely, (i) Biomedical Metadata Quality assessments and (ii) Methods that help in identifying biomedical terms in the text – here we discuss different methods for identifying concepts/terms in the text.

2.1 Biomedical metadata quality assessment

There exist several projects for assessing the quality of biomedical metadata, the methods range from using ontologies, clustering to topic modeling. In [18], a survey is presented regarding the use of controlled vocabularies when defining metadata categories in BioSamples¹. The analysis established that the quality of metadata in BioSamples is poor because of a lack of structured format and vocabularies to describe it. Furthermore, they add that there should be a robust method to enforce authoring of metadata. However, no further analysis was performed regarding improving the quality of existing metadata.

In [28] and [61], gene expression metadata quality assessment was performed on the Gene Omnibus Expression (GEO) database. The assessment was performed using (a) clustering methods and (b) crowdsourcing (i.e. non-expert human workers). In the former, the metadata (keys) were clustered based on (a) lexical similarity, (b) core concepts and (c) value similarities. In the latter empirical analysis was performed on the same set of keys using crowdsourcing by submitting microtasks on the Figure Eight² platform. While both methods were able to classify keys that contained the category term (e.g. ‘disease state’ in the category ‘disease’), the clustering algorithm misclassified certain keys (e.g. ‘stage’ in the category ‘age’) and there was low consensus amongst workers for key that could belong to more than one category (e.g. ‘disease specific survival years’ that was categorized either into the ‘disease’ or ‘time’ category).

¹<https://www.ncbi.nlm.nih.gov/biosample/>

²<https://www.figure-eight.com/>

In [47], GEO data is used to predict unstructured metadata using the information from the unstructured elements in the metadata. Here the authors employ topic modeling on the unstructured elements of the metadata and use these topics as features to train a supervised classifier namely Support Vector Machines (SVM). Furthermore, a similar classification as written above is performed but in this case, the features were extracted using term frequency-inverse document frequency (TF-IDF). When comparing the results, the feature set using TF-IDF performed better than the features using topic modeling. Although, this method can be useful when predicting structured metadata but it is not very helpful when used over a very large dataset with over 70,000 features per document.

Similarly in [43], prediction of metadata was done using existing metadata, with the help of association rule mining in addition to supervised machine learning. A subset of metadata elements (namely organism and molecule) were selected to perform the predictions. The limitation in this approach was the information gets lost because of arbitrary thresholds of rule mining which further affects the performance of the classification.

In the work [42], the authors develop a recommendation engine to facilitate the submission of metadata with their use case also being GEO. When users are submitting their metadata, a list of suggestions is generated using context of the information queried. This recommendation also takes into account the previously entered values. These recommendations would reduce the effort of typing when submitting metadata, which would improve the quality of metadata by making it less error-prone, comparable, reusable and interoperable.

2.2 Identifying biomedical terms in text

Here we discuss different methods that are used in identifying terms in biomedical text and methods which are used to cluster similar terms together.

2.2.0.0.1 Topic Modeling In our previous work [39], we used topic modeling to identify meaningful topics of the biomedical metadata for BioSamples dataset³. Can we use topic modeling to identify meaningful topics in biomedical metadata? The metadata in BioSamples is the form of key: value pairs. We applied topic modeling, specifically NMF [36] (non-negative matrix factorization) to the keys and values separately. We divided the keys into two types: (i) lexically similar keys and (ii) non-lexically similar keys. Similarly, we tried to divide the values into two parts: (i) lexically similar values and (ii) non-lexically similar values, but we did not work with the non-lexically similar values because of the heterogeneity present in the values, which made it difficult for the NMF to group topics together in a meaningful way. On the contrary, when working with the keys, the topics were representative of the keys. Although there were limitations to this method as well. The algorithm does not output keywords

³<https://www.ebi.ac.uk/biosamples/>

that are less frequently occurring since it uses TF-IDF to predict the topics. This leads to losing out on important information. We propose to tackle this by adding weights on the less frequently occurring keywords as input.

2.2.0.0.2 Natural Language Processing (NLP) In [9], the authors experiment with different models namely skip gram, Continuous Bag of Words (CBOW) to create word embedding for Biomedical NLP on the texts of PubMed⁴, PMC⁵. They also test their model by tuning certain hyperparameters namely negative sampling, subsample rate, min-count, learning rate, vector dimension, context window size the authors use a large input to create word embedding for biomedical NLP. They also mix and match their models with different types of text preprocessing namely normal text, sentence-shuffled text, and lower-case text. They conclude that preprocessing using sentence shuffling for PubMed texts perform better than a larger text corpus of the PubMed Central Open Access subset (PMC). They also conclude that the difference in performance after tuning the hyperparameters can be debated over.

2.2.0.0.3 Named Entity Recognition (NER) and Deep Learning In [22] the authors argue that deep learning with word embeddings would improve the NER in the biomedical domain. They present two methods (i) a traditional method of NER - a basic conditional Random Field (CRF) and (ii) a deep learning network namely called long short-term memory network-conditional random field (LSTM-CRF) and compare the results. They select five different entity classes namely (i) Chemicals, (ii) Genes/Proteins, (iii) Species, (iv) Diseases and (v) Cell lines for their evaluation. They select a set of 24 corpora for these aforementioned five entity types. They use three different word embeddings of Patents, PMC-PubMed and Wiki-PMC-PubMed. These methods perform best with the word embedding of Wiki-PMC-PubMed because it contains information of both domain specific(PMC and PubMed) and general (Wikipedia). Overall it is observed that the deep learning method performs better than the baseline CRF method, which is interesting because the model does not know from before what entity types it is dealing with. Lastly, they conclude that for different methods for various entity types can be expensive, which is why a common tool like this should be used for biomedical NER.

In [55] the authors use deep learning to create a scalable NER model for text data in the biomedical domain using the entity-free or key-value pairs of the BioSamples dataset. The motivation behind this work was to facilitate information retrieval and extraction. They extract all the metadata from the BioSamples and train a Bidirectional Long Short term Memory (BiLSTM) neural network with the help of word embeddings developed by [9] of Wiki-PMC-Pubmed. They compare their results from existing projects like MetaMap⁶. With the help of word embeddings, they were able to group various types of

⁴<https://www.ncbi.nlm.nih.gov/pubmed>

⁵<https://www.ncbi.nlm.nih.gov/pmc>

⁶<https://metamap.nlm.nih.gov/>

free-text annotations at word, sentence and entity level. In this approach the author’s group the free-text present in the metadata of the dataset.

2.2.0.0.4 Machine Learning In one of our previous projects [40], we used a machine learning approach to perform an assessment of biomedical metadata. The dataset used was a crowdsourced gold standard set from [61]. We used this as a gold standard while performing our experiment. We selected 11 key types as they were top frequently occurring in GEO: (i) tissue, (ii) age, (iii) gender, (iv) strain, (v) cell type, (vi) treatment, (vii) cell line, (viii) sex, (ix) disease, (x) genotype and (xi) time. We used topic modeling to create features and used the scores generated from the topic model as weights. The binary classification approach got the highest accuracy of 0.98 for the key type ‘genotype’, whereas for a multi-class classification approach we got an accuracy of 0.40. The limitations of our approach were that it did not support the multi-label classification technique and the size of data was less.

In this thesis project, the innovative approach is to use scientific biomedical text to predict the metadata which would be a combination of the methods that are described above.

Chapter 3

Preliminaries and Dataset

3.1 Preliminaries

3.1.1 RDF, Ontology, and SPARQL

The Resource Description Framework (RDF) is a framework to represent data in the web [56]. It is an XML based language to represent the information that is present on the web. RDF contains basic three elements: Property, Resource, and Statement (subject, predicate, and object). For example a statement “Stuti knows Tom Hanks” would translate to `uri://people#StutiNayak25 http://xmlns.com/foaf/0.1/knows uri://people#TomHanks62`.

W3C defines an ontology as “the terms used to describe and represent an area of knowledge” [25]. An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them [41]. Everyone has their own definitions when it comes to ontology, another definition given by [20, 21] - “from an AI point of view, an ontology is defined as the *explicit specification of organization*” which is widely accepted in the AI community. It is further explained that an ontology should be considered as an agreement regarding *shared conceptualizations*. Shared conceptualizations include conceptual frameworks for modeling domain knowledge; content-specific protocols for communication among inter-operating agents; and agreements about the representation of particular domain theories. In the knowledge sharing context, ontologies are specified in the form of definitions of representational vocabulary. A very simple case would be a type hierarchy, specifying classes and their subsumption relationships.

Lastly, when we want to extract information that we want using the RDF data we need a query language there we use SPARQL [23]. It can also be used to manipulate data stored in RDF format. It supports various result formats namely: the Extensible Markup Language (XML), the JavaScript Object Notation (JSON), Comma Separated Values (CSV), and Tab Separated Values (TSV).

3.2 Dataset

As mentioned earlier in Chapter 1 for re-using we need to understand the meta-data. An example of a GEO sample is shown in the Figure 3.1. The highlighted

NCBI GEO Accession Display

Scope: Self Format: HTML Amount: Quick GEO accession: GSM549326

Sample GSM549326 Query DataSets for GSM549326

Status	Public on Aug 11, 2010
Title	WB-Still-5163
Sample type	RNA
Source name	Human whole blood
Organism	Homo sapiens
Characteristics	age: 83 gender: Male ethnicity: Caucasian illness: Still
Extracted molecule	total RNA
Extraction protocol	Ambion Temput Isolation kit
Label	biotin
Label protocol	Biotynlated cRNA were generated following manufacturer protocols using the Illumina TotalPrep Amplification Kit from Ambion.
Hybridization protocol	Hybridized 12-Sample Beadchip according to manufacturer protocol
Scan protocol	SentrixScanVersion 2.3.0.13 (This information is in the .XML file)
Description	Whole blood was collected in Tempus tube
Data processing	The data were analyzed using BeadStudio version 1.5.1.3 using the average chip normalization method.
Submission date	Jun 02, 2010
Last update date	Aug 11, 2010
Contact name	Damien Chaussabel
E-mail	DChaussabel@benaroyaresearch.org
Organization name	Baylor Institute for Immunology Research
Street address	3434 Live Oak
City	Dallas
State/province	TX
ZIP/Postal code	75204
Country	USA

Figure 3.1: An example of GEO Sample

part is the field characteristics. The field is structured as key: value pairs where “age” is the key and the number 83 is the value. We chose the “characteristics” field since it is semi-structured. Examples of good metadata and bad metadata in presented in Figure 3.2 and 3.3. From the Figure 3.2, in the field about gender it is filled with the number 2 - which is semantically inaccurate as it is unclear what the 2 refers to. Thus it is difficult to understand what that Sample was about. Similarly, there is no way a human being lives for 160 years, but in the metadata *age: 160* is mentioned. If someone wants to understand what this information is about except that it is something about human beings, it would not be a very trivial task. On the other hand, from the Figure 3.3, one can make out the experiments involved human beings and about leukemia.

The scientific publications contain a lot of information about the experiments done, this information would be useful to determine good metadata. Therefore

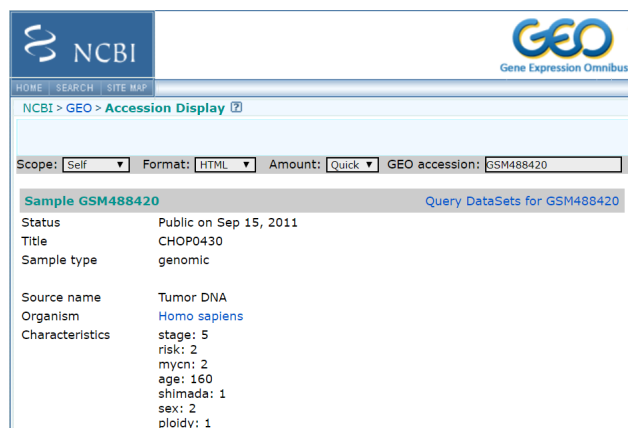


Figure 3.2: Example of a bad metadata

for predicting metadata, the first step is to look for a gold standard dataset which would have annotated terms of metadata elements. We first start with the metadata category *disease*, for that we need to have a dictionary which can be used for predicting disease values. Therefore, for this purpose, we use the NCBI Disease Corpus [14]. This corpus identifies disease terms using 14 human annotators in 793 scientific publication abstracts and titles from PubMed ¹, which serves the purpose as when we test the model with a known scientific text we would know how accurate are the predictions. This corpus identifies disease terms and links it to ontology namely Medical Subject Headings (MeSH)² and Online Mendelian Inheritance in Man (OMIM)³.

The disease names identified are classified into the following four concepts:


- DiseaseClass: A string which represents a family of diseases, then it was annotated to a disease class. For example: “autosomal recessive disease” is a *Disease Class* category.
- SpecificDisease: A string that does not have any children class. For example: “Diastrophic dysplasia” is a *Specific Disease*.
- Modifier: A string that is a disease name but it might modify a sentence containing a noun. In simple words, if a disease name occurs which is a secondary and tries to focus on any other disease is annotated as a *Modifier*.

For example: In the abstract text “Although this mutation was initially detected in four of 33 colorectal cancer families analysed from eastern England, more extensive analysis has reduced the frequency to four of 52 English HNPCC kindreds analysed.”

¹<https://www.ncbi.nlm.nih.gov/pubmed/>

²<https://www.nlm.nih.gov/mesh/>

³<https://www.omim.org/>



BioSamples

[Home](#)
[Search](#)
[Submit](#)
[Help](#)
[About BioSamples](#)

BioSamples > SAMN00172176

Sample SAMN00172176

Name	Homo sapiens leukaemia cells RT-PCR
Description	1 Homo sapiens samples
Release date	2015-07-10
Last updated	2015-07-25
Submission title	1 Homo sapiens samples
Submission identifier	GNC-SAMN00172176
Organism	Homo sapiens
Cell type	primary leukaemia cells
Model	Generic
Package	Generic.1.0
Synonym	LIBEST_012622

Figure 3.3: Example of a satisfactory metadata

The experts are asked to annotate in this way: “colorectal cancer” as Modifier category and “HNPCC” as Modifier category.

- CompositeMention: If more than one disease names occur together then that string was annotated to *Composite Mention*.

For example, a protein required for ubiquitin-mediated degradation of beta-catenin, but a small percentage of colon and some other cancers harbour beta-catenin-stabilizing mutations.

The Annotation is - “colon and some other cancers”: CompositeMention

An example of annotation is shown in Listing 3.1. The number 10192393 is the PubMed Identifier, the string “[t]” denotes the title and “[a]” denotes the abstract of the scientific publications. All the terms below the abstract are the identified disease terms such as skin tumour, cancer etc. The terms “D012878”, “D009369” are the unique identification number of skin tumour and cancer respectively from the MeSH ontology.

```

1 10192393|t|A common human skin tumour is caused by activating mutations in
   beta-catenin.
2 10192393|a|WNT signaling orchestrates a number of developmental programs. In
   response to this stimulus, cytoplasmic beta-catenin (encoded by CTNNB1)
   is stabilized, enabling downstream transcriptional activation by members
   of the LEF/TCF family. One of the target genes for beta-catenin/TCF
   encodes c-MYC, explaining why constitutive activation of the WNT pathway
   can lead to cancer, particularly in the colon. Most colon cancers arise
   from mutations in the gene encoding adenomatous polyposis coli (APC), a
   protein required for ubiquitin-mediated degradation of beta-catenin,

```

```

3   but a small percentage of colon and some other cancers harbour beta-
4   catenin-stabilizing mutations. Recently, we discovered that transgenic
5   mice expressing an activated beta-catenin are predisposed to developing
6   skin tumours resembling pilomatricomas...
7   10192393      15      26      skin tumour      DiseaseClass      D012878
   10192393      443     449      cancer      DiseaseClass      D009369
   10192393      483     496      colon cancers      DiseaseClass      D003110
   10192393      539     565      adenomatous polyposis coli      SpecificDisease
   D011125
   10192393      567     570      APC      SpecificDisease      D011125

```

Listing 3.1: An example of annotation

As mentioned earlier, the corpus contains in total 793 PubMed titles and abstracts which is divided into three sets (i) training, (ii) testing and (iii) development. The Table 3.1 shows details of the corpus:

Table 3.1: NCBI disease Corpus characteristics

Corpus Characteristics	Training	Development	Testing
PubMed Citations	593	100	100
Disease Mentions	5,148	791	961
Specific Disease	2,959	409	556
Disease Class	781	127	121
Modifier	1,292	218	264
Composite Mention	116	37	20

The corpus is annotated by fourteen domain experts. There were two-annotators per document which were paired randomly. The annotation was completed in three phases and was checked for consistency throughout the whole corpus as shown in the Figure 3.4. In total, we have 2136 diseases. The statistics of the annotations are as follows are shown in Table 3.2.

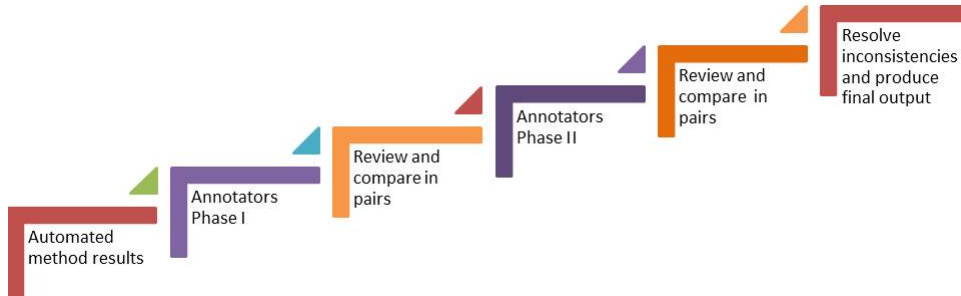


Figure 3.4: Annotation Process (Source: <https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/DISEASE/>)

3.2.0.0.1 Data Preprocessing In the dataset, we have a set of scientific publication titles and abstracts in addition to the identified disease terms and

Table 3.2: Input corpus statistics

Target Labels	Count
CompositeMention	112
DiseaseClass	571
Modifier	222
SpecificDisease	1231
Grand Total	2136

respective disease concepts from the ontology as we saw in the Listing 3.1. Since a neural network model cannot process raw text data, there is a need to perform some preprocessing steps. Firstly, the data is cleaned by removing all the (i) punctuation (filters='! "\$%&()*+,-./:;<=>?@[\\]^_`'), (ii) numbers and (iii) convert all the text into lowercase. For training the model, we only use (i) the abstract, (ii) the disease names and (iii) the Unique ID of these diseases from the MeSH ontology. The abstract is used because it contains more information when compared to the title. After that, the text is separated from the disease names as we treat the disease names as the labels that would be identified using the text by the model. We get the following characteristics of the abstracts:

- max length: 450
- mean: 191.1640826873385
- min length: 32
- median: 190.0

3.2.0.0.2 Text Preprocessing The text is tokenized - converted into a list of words, and fit according to the size of the text, this converts the text into a set of words which could be called tokens. Then these tokens are integer encoded - given a unique index, the highest number being the size of the words. Then we convert these words into a sequence which gives the words a unique identification number. Then we need to pad this sequence which means to normalize the sequences into a maximum length so that the neural network knows what length of the sequence to expect. For example - some sentences will be less than 200 let's say 155, so we insert 45 zeros to make it 200. Some might be bigger say 220, so we remove last 20. We test the model with varying sequence lengths.

Chapter 4

Methods for Text Classification

4.1 Introduction

The problem of text classification or text categorization poses the following problem: assign a set of predefined categories to unstructured text that can be used to organize and structure the text appropriately according to the use case. Text classification is a pivotal problem with several applications in biomedicine. For instance, problems such as recognizing reportable cases of cancer from pathology reports, identifying certain phenotypes from clinical notes, performing word sense disambiguation (that is, given a context determine the semantic meaning for the usage of an ambiguous word), and associating medical subject headings (MeSH terms) to scientific articles, can all be reduced to instances of generic text classification problem. Further, text classification can be grouped into two categories namely, (i) multiclass classification (i.e. labels are mutually exclusive) and (ii) multilabel classification where each input can be assigned to more than one label. By definition, it is clear that multiclass classification is a special case of multilabel classification and hence the latter problem becomes more harder to solve than the former.

Traditionally, the problem of text classification is solved by leveraging discriminative models (models that model decision boundary by estimating posterior probabilities) such as support vector machines (SVMs) and logistic regression (LR), only to mention a few, trained on features extracted from the text including bag of words (BOWs) [51]. Moreover, it has been observed in the literature that performance gains are achieved with better feature selection techniques, underlying dataset selection, and ensemble approaches [65]. On the other side, better performing models have also be devised by leveraging the field of feature engineering, where the objective is to derive domain specific features that are more relevant and important. For instance, emoticons, exclamations, and hashtags form important features for sentiment analysis of Twitter feed [33].

Moreover, especially in the field of biomedicine, exploiting interoperable concept explanations and inter-concept relations from domain specific knowledge base such as unified medical language system (ULMS)¹ have led performance gains in the accuracy of the underlying models [59]. To this end, models with linear classifiers with ensemble modeling and domain specific feature engineering have the capacity to attain competitive results for the task of text classification.

We perform two experiments for the problem of text classification: (i) Multi-label classification and (ii) Deep learning - specifically CNN and BiLSTM. We illustrate a comparison of the performance of the aforementioned methods in Chapter 5. The motivation behind this comparison of employing two techniques is to answer the research question as to which method performs better when we deal with the prediction of metadata (Refer RQ4 in Chapter 1).

4.2 Multi-label Classification

The number of labels (or categories) entails the different kinds of classification problems. For example: if the number of labels $|L| = 2$ then the problem is called binary classification, for $|L| > 2$ labels the problem becomes a multi-class classification. The problem of multi-label classification similarly deals with more than one label, but each label is considered as a different classification problem, but these problems are somewhere related to each other. On the contrary, in a multi-class classification, the labels are mutually exclusive [54]. For instance, in a multi-class problem would consider that a scientific publication would only talk about one out of several diseases. Whereas in a multi-label classification the scientific publication might talk about several diseases or none diseases at all. In order to use the traditional classification techniques like Random Forest Classifier, Decision trees and so on, the problem needs to be transformed in a way that these techniques can be used. In this project, we use three transformation techniques (i) Binary Relevance, (ii) Label Powerset and (iii) Classifier Chains.

4.2.1 Problem Transformations

Here we briefly describe the problem transformations where traditional classification techniques are modified in such a way that they can be used to perform multi-label classification.

4.2.1.0.1 Binary Relevance (BR) This transformation converts the multi-label problem into a single label problem for each label. Once these problems are solved, a union is taken for all the classes which is the output of the problem. In simpler words, this problem solves a binary classification for each label with any classifier picked as seen in Figure 4.1. The major disadvantage in this approach is that it does not take label dependencies into account.

¹<https://www.nlm.nih.gov/research/umls/>

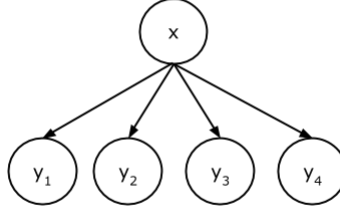


Figure 4.1: Multi-label problem transformation - Binary Relevance

4.2.1.0.2 Label Powerset (LP) In this problem transformation, a multi-label problem is converted into a single multi-class problem which can take many values. It takes the labels y_1, y_2, \dots, y_n (refer Figure 4.2) and makes several combinations of these labels and solves a multi-class problem for every such combination. This method computes a large space of classes. In total it solves $2^{|L|}$ such problems where $|L|$ is the number of labels.

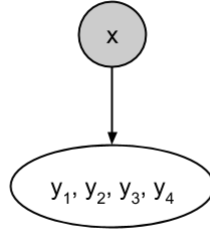


Figure 4.2: Multi-label problem transformation - Label Powerset

4.2.1.0.3 Classifier Chains (CC) Classifier chains [13, 49] are an improvement in the BR transformation, which takes into account the dependency between the labels if there exists any. Let's say the feature set is: $(x_1, x_2, x_3, \dots, x_n)$ and the set of labels is: $(y_1, y_2, y_3, \dots, y_n)$. The initial feature set is used to predict y_1 , after that $(x_1, x_2, x_3, \dots, x_n, y_1)$ is used to predict y_2 . At n^{th} step, feature set $(x_1, x_2, x_3, \dots, x_n, y_1, y_2, y_3, \dots, y_{n-1})$ predicts y_n (refer Figure 4.3). The order of prediction can be tuned in order to get better results. For a large set of labels this method is slow as it takes $n!$ combinations of orderings.

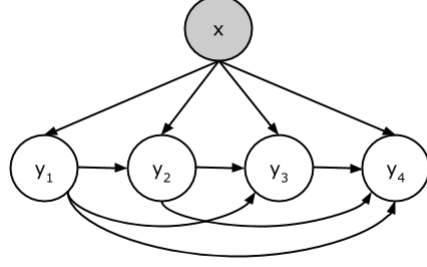


Figure 4.3: Multi-label problem transformation - Classifier Chains

4.2.2 Problem adaptation and Classifiers

Here we discuss the adaptation of existing classification algorithms so that they have the ability to perform multi-label classification. Furthermore, we discuss the standard classifiers that have been used.

4.2.2.0.1 Multi-label k Nearest Neighbours (MLkNN) Another adaptation of k nearest neighbours(kNN) for multi-label classification was done by [63]. In kNN assigns the most commonly occurring neighbours among the calculated k ‘nearest neighbours’. Whereas in MLkNN, the labels assigned are the most common labels in the neighbourhood of k nearest neighbours. In simpler terms, given a test set, MLkNN uses kNN to find nearest examples and uses Bayesian Inference to predict the labels.

$$y_j = \begin{cases} 1, & \text{if } P(c_{j,x}|y_j = 1)P(y_j = 1) \geq P(c_{j,x}|y_j = 0)P(y_j = 0) \\ 0, & \text{otherwise} \end{cases} \quad (4.1)$$

(where $c_{j,x} :=$ the number of number of examples present in the neighbourhood of x with $y_j = 1$; Probabilities calculated from the training data)

4.2.2.0.2 Decision Trees (DT) A decision tree use tree structure perform classification. In such a tree (i) each interior node tests one attribute, (ii) every branch corresponds to an attribute and (iii) each leaf node is the labeled with the class i.e. label/category. A classification using decision tree is presented in Listing 4.1. It is a strategy for approximating discrete-esteemed capacities that are powerful to load information and equipped for learning disjunctive expressions. The group of decision tree learning calculation incorporates generally utilized calculation, for example, ID3 [48], CART [8] and C4.5 [58]. These decision tree learning techniques seek totally expressive speculation space and

hence keep away from the troubles of limited theory spaces. Their inductive inclination is an inclination for little trees over substantial trees.

```

1  Classify(x: instance, node: variable containing a node of DT)
2      if the node is a classification node then
3          return the class of node;
4      else
5          determine the child of the node that match x.
6          return Classify(x, child).

```

Listing 4.1: Classification with Decision Trees

4.2.2.0.3 Random Forest A random forest is an information build connected to machine learning that grows substantial quantities of random decision trees dissecting sets of factors [7]. This sort of calculation improves the manners in which that advances examine complex information.

All in all, decision trees are famous for machine learning assignments. In a random forest, engineers build sets of random decision trees to all the more cautiously detach information from information mining, with various connected variable exhibits. One approach portraying the theory behind the random forest is that since the random trees have some cover, specialists can assemble frameworks to consider information needlessly with the different trees and search for patterns and examples that help a given information result. For instance, if five random trees give data on a similar variable from a subset, and four of them concur, the machine learning calculation may use that "dominant part vote" to construct models dependent on probabilities. In a wide range of sorts of machine learning, develops like the random forest can assist mechanical frameworks with drilling down into information and give increasingly modern investigation.

4.2.2.0.4 Multi-layer perceptron (MLP) Although neural networks are also explained in Section 4.3.1, we define MLP here as we use it as a baseline classifier. Multilayer feedforward networks, an essential class of neural networks [24]. Typically, the system comprises of a lot of tangible unit (source hubs) that establish the info layer, one or increasingly concealed layers of computational nodes, and a yield layer of calculation nodes. The input flag engenders through the system in a forward direction, on a layer-by-layer basis. These neural systems are regularly alluded to as multilayer perceptrons (MLPs), which speak to speculation of the single-layer perceptron.

Multilayer perceptrons have been connected effectively to tackle some troublesome and differing issues via preparing them in an administered way with a profoundly prominent calculation known as back-engendering calculation. A multilayer perceptron has three particular qualities:

1. The model of every neuron in the system incorporates a nonlinear enactment work.
2. The system is comprised of at least one layers of concealed neurons that are not part of the information or yield of the network. These shrouded

neurons empower the system to learn complex assignments by separating dynamically progressively significant highlights from the information design.

3. The system displays high degrees of connectivity, determined by the neural connections of the system.

It is through the mixture of these attributes together with the capacity to gain as a matter of fact through preparing that the multilayer perceptron determines its registering power.

4.3 Deep Learning

The advent of deep neural networks (deep nets) in the last decade or so has led to a foundation for generic alternatives to supervised learning, especially for the task of object classification. Deep nets eliminate the laborious process of feature engineering and automatically learn high level representations of input which suites best for the underlying classification problem. Although the resurgence of deep nets was initially meant for the field of computer vision, recently it has also been applied to natural language processing tasks (NLP)[2, 12, 37] especially through learning distributed representations of words as vectors in high dimensional space. These vectors help the model in guiding elementary tasks such as part-of-speech (POS) tagging and parsing as well as abstract tasks such as text classification and machine translation. Typically, the novel deep learning approaches for text classification rely on architectures based on convolutional neural networks (CNNs) or recurrent neural networks (RNNs) [60]. In Figure 4.4, you can see a typical deep learning based text classification architecture pipeline.

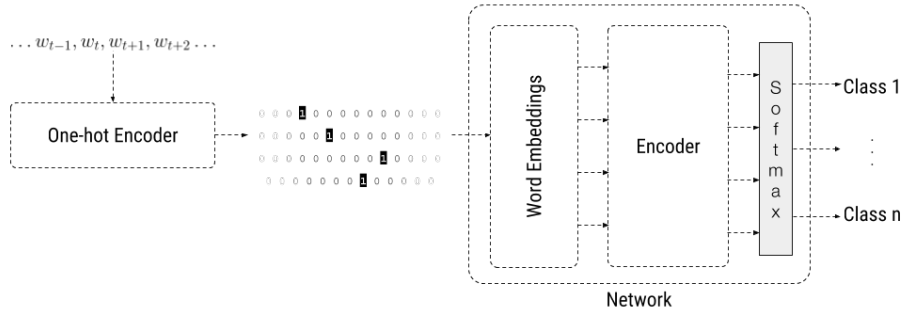


Figure 4.4: Typical neural network based deep learning architecture for multi-class classification of texts.

Now we unfold each and every component of the architecture in Figure 4.4.

4.3.1 Unfolding the architecture

For an average human, understanding the inherent meaning of the words based on the context they are used, is an easy task. In the same spirit, a deep learning model needs a clever representation of text segments so that they can understand the semantic similarity of related words and also differentiate each word from others.

4.3.1.0.1 One-hot Encoding Assigning unique, discrete IDs to the word tokens, that is, represented by a *one-hot encoding vector* helps in understanding the model to differentiate between two different words. This, in turn, has an advantage as neural networks expect vectors as input. Although, this still does not provide any useful information to the model about the relationship between the words and in fact additionally, leads to *sparsity*. To state an example, treating words as discrete atomic symbols, for instance, representing a *cat* by *Id100* and *dog* by *Id150* does not provide any meaningful information to the model regarding the relationship that may exist between the two symbols as they are some arbitrary encodings. This amounts to the model leveraging very little of what it has learned about *cats* when it is processing data about *dogs* (such that they are both animals, four-legged, pets, only to mention a few).

4.3.1.0.2 Word Embeddings To overcome the problems regarding sparsity and semantic similarity of related words in one-hot encoding, [2] developed the concept of word embeddings back in 2003. Despite being introduced before, word embeddings are still relevant today and highly active research domain in deep learning [4, 30, 29, 46]. The concept relies on the *Distributional Hypothesis* of J.R. Firth (1957) that states that “*You shall know a word by the company it keeps*”, that is, words that appear in the same contexts share semantic meaning. As we are in the context of neural networks, we are only concerned with predictive methods (i.e. neural probabilistic language models). Predictive models directly try to predict a word from its neighbors in terms of learned small, dense embedding vectors (considered parameters of the model).

A word embedding $\mathcal{W} : \text{words} \rightarrow \mathbb{R}^n$ is a parameterized function that maps words to high dimensional vectors (typically between 200 to 500 dimensions). In other words, it is a dense vector space that forces a fixed dimension on the vector space and forces similar words to appear nearby each other across some dimensions. Typically, the function \mathcal{W} is a look-up table, parameterized by a matrix θ where each row corresponds to an embedding for a particular word: $\mathcal{W}_\theta(w_n) = \theta_n$ (refer Figure 4.5). \mathcal{W} is initialized randomly by having random row vectors for each word and the model learns to capture meaningful syntactic and semantic regularities in order to perform some task. Word2vec [37] and GloVe [45] are the two most popular word embedding models to obtain off-the-shelf trainable word embedding matrices. Moreover, word embeddings exhibit an interesting property, namely, it encodes semantic components as linear vector differences. (refer Figure 4.6).

$$\begin{array}{c}
\text{"cat"} \\
[0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0]
\end{array}
\times
\begin{bmatrix}
0.21 & 0.42 & 0.16 & 0.86 & 0.67 \\
0.86 & 0.90 & 0.20 & 0.38 & 0.19 \\
0.49 & 0.24 & 0.87 & 0.95 & 0.88 \\
0.80 & 0.64 & 0.19 & 0.32 & 0.38 \\
0.71 & 0.54 & 0.66 & 0.61 & 0.40 \\
0.39 & 0.17 & 0.19 & 0.54 & 0.41 \\
0.17 & 0.30 & 0.32 & 0.96 & 0.76 \\
0.05 & 0.32 & 0.80 & 0.92 & 0.81
\end{bmatrix}
= [0.71 \ 0.54 \ 0.66 \ 0.61 \ 0.40]$$

Figure 4.5: The figure illustrates how to obtain embedding of a particular word “cat” from a word embedding matrix. The one-hot encoding vector of the word “cat” is multiplied to the word embedding matrix to obtain the embedding vector for *cat*.

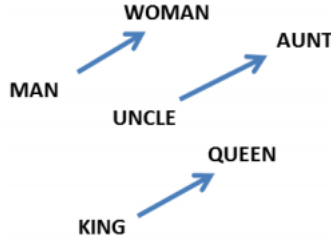


Figure 4.6: This figure illustrates the property of word embeddings where linear vector differences between male-female vector is preserved, that is, constant. That is, $\mathcal{W}(\text{“woman”}) - \mathcal{W}(\text{“man”}) \simeq \mathcal{W}(\text{“aunt”}) - \mathcal{W}(\text{“uncle”}) \simeq \mathcal{W}(\text{“queen”}) - \mathcal{W}(\text{“king”})$ (Image Source [37]).

4.3.1.0.3 Encoder Once we have the word embedding matrices from the raw text, in order to perform classification tasks, commonly used encoder architectures are convolutional neural networks (CNN) and recurrent neural networks (RNN). Next, we briefly summarize how is the classification task performed using the aforementioned architectures.

4.3.1.0.4 Neural Networks The neural network (NN) approach is widely used in the machine learning and deep learning algorithms to solve problems with complex, sparse data [35]. The word ‘neural’ is used because they are loosely inspired by neuroscience. The goal of such a network is to estimate a function f^* . In a traditional classifier $y = f^*(\mathbf{x})$, it maps the input \mathbf{x} to output category y . Whereas in the feedforward neural network the model defines a mapping $y = f(x; \theta)$. It then learns the entries of the parameter in θ which is in-turn best approximation of the function. The basic computational unit of a NN is a neuron (refer Figure 4.7) which is often referred to as a node or a unit. It receives input from other units or from an external source after which it computes an output (see Figure 4.8). Each output from this neuron consists of an associated weight (w), which is calculated according to the relative

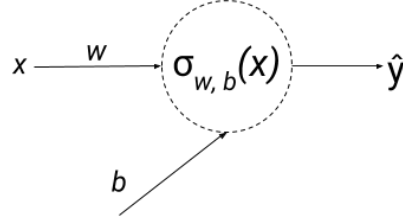


Figure 4.7: Typical neuron in a neural network, which can be intuitively thought of a building block of the underlying neural network.

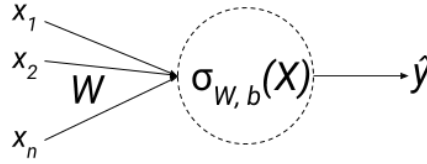


Figure 4.8: Vector inputs in the neuron

importance with respect to other inputs. The node applies a weighted function to the sum of inputs.

The intention behind the network is to learn these set of weights and to be able to control the strength and the influence of these parameters according to the problem that is being solved using the NN approach. We control such an effect of the parameters using an activation function which should be associated with a neuron. Using this function, the node computes the output given a set of inputs. One such activation function is Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$. A parameterized sigmoid function is $\sigma_{w,b}(x) = \frac{1}{1+e^{-wx+b}}$. The advantage of using a non-linear activation function such as sigmoid is that these networks are able to solve non-trivial problems using a relatively smaller number of parameters. There are also other types of non-linear activation functions such as hyperbolic tangent (\tanh), Rectified Linear Unit ($ReLU$).

In a neural network architecture, there are layers which are classified into three types which are mentioned below, all these layers contain nodes which are interconnected to each other in a dense manner also shown in Figure 4.9:

- **input layer** - this brings in the data into the architecture which is then passed on to the next layer for processing
- **hidden layer(s)** - after the input layers, there are a certain number of interconnected layers these can be specified accordingly. Here, the network takes a bunch of weighted sets and use an activation function to emit the output.
- **output layer** - the last layer of nodes where we get the output.

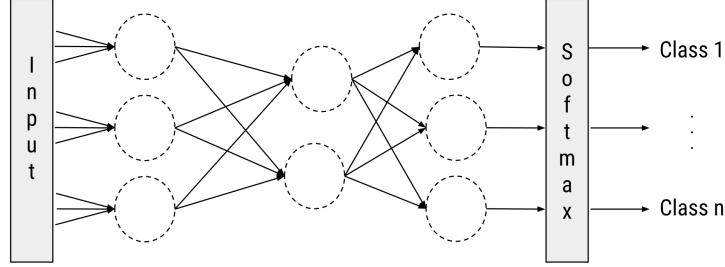


Figure 4.9: A simple neural network architecture for the task of multiclass classification, where the output layer is followed by a softmax layer, that takes outputs from the output layer and converts a distribution of numbers into probabilities.

While training the neural networks we need to see how every parameter effects on the performance. For this purpose, we calculate the model performance using the function known to us as *loss function*. A loss function is a difference between the model output (\hat{Y}) and the given label (Y) for a data point. For instance, one such loss function is mean square error method and is written as,

$$L(\hat{Y}, Y) = \frac{1}{2n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (4.2)$$

Given enough hidden units (neurons), a two-layered neural network that can approximately imitate any continuous function, over a wide range of activation functions. Although the challenge is to find the right values of the parameters given the input data. This is where backpropagation comes into practice. We know from high school calculus that for a function $y = f(x)$, $f'(x)$ or $\frac{dy}{dx}$ is an indicative of modifying the value of x , where the objective is to minimize $y = f(x)$. That is, in order to reduce $f(x)$, we have to modify x in small steps with an opposite direction of the sign of $f'(x)$ —this is gradient descent. Backpropagation is done using gradient descent. The gradients are calculated w.r.t. model parameters for each parameter. The parameters are modified in such a way that the parameters point in that direction where the loss is decreasing. The parameters (weights) are modified using the following formulation, where η (positive scalar) is a hyperparameter known as learning rate of the neural network model (determines how big a step to take in the direction opposite to the gradient)

$$w_n^{(t+1)} = w_n^{(t)} - \eta \frac{dL}{dw_n^{(t)}} \quad (4.3)$$

Summarizing from Figure 4.10, we give the input data x to the neural network, y is the input label of the data and $f_W(x)$ is the model which gives the output \hat{y} ($\hat{y} = f_W(x)$). After this loss is calculated: $L(\hat{y}, y)$. This is the forward propagation. Now in the backward propagation: we calculate gradients $\frac{dL}{dW_{1:n}}$. Then the parameters are updated using the Equation 4.3 in each epoch.

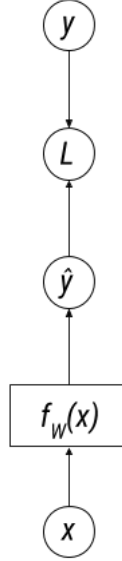


Figure 4.10: A forward and backward propagation in the neural network.

4.3.1.1 CNN

Convolutional neural networks are a special kind of neural networks which employ a grid-like operation. The name “convolution” is used because it uses the mathematical operation convolution. In simple words: in a CNN convolution operation is used in at least one of the layers in place of general matrix multiplication [19]. They are widely used in the context of images [34] where images are considered 2D grid of pixels. Although, the architecture can be used if sentences or paragraphs or documents can be represented as matrices or vectors, as it is the case in images. Word embedding matrices typically act as pixel matrices of images in the context of text classification. One of the reasons as to why CNNs are successful is because of its inherent property of keeping a number of copies of the same type of neurons. CNN has been successfully applied in NLP tasks [15, 62].

From the network point of view, CNN’s are essentially several layers of convolutions (refer Figure 4.11) stacked upon each other with non-linear activation functions applied to the output of neurons in the same layer. A typical layer in a CNN has three stages: (i) convolution operation (refer Figure 4.13), (ii) a non-linear activation like a rectified linear unit and (iii) a pooling function which lists the set of output with a certain statistical explanation of outputs of the neighbourhood. These three stages accommodate in the standard feed-forward architecture of linear and non-linear activations. As we can see from Figure 4.11, the first stage is linear and the second layer when combined with

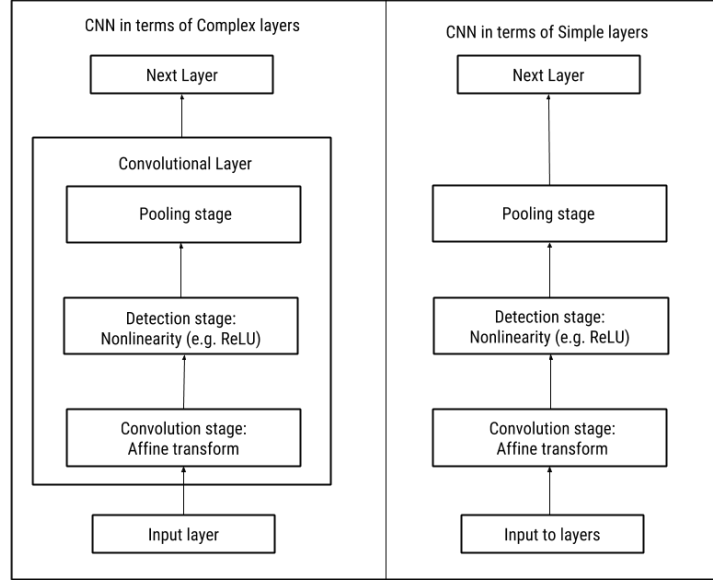


Figure 4.11: The figure illustrates the components of a typical convolutional neural network (CNN). The figure in the left views CNN as a small number of complex layers, where within each layer a number of operations are applied. While on the right, CNN is viewed as a larger number of simple layers performing elementary operations.

the third stage form a non-linear activation. One of the most used pooling statistics is *max-pooling*. Most commonly, max-pooling is used as a downsampling technique with, for instance, say 2×2 dimension filter and stride equal to 2 (Figure 4.12), which reduces the representation size and decreases the computational and statistical burden on the next layers.

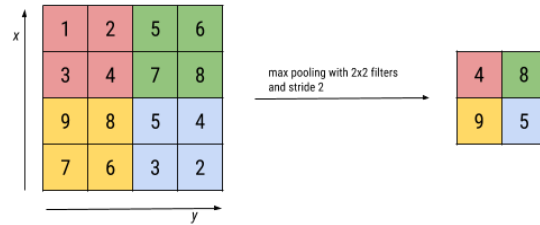


Figure 4.12: This figure illustrates the operation of max-pooling to summarize the feature maps.

Figure 4.13 illustrates that convolution is a binary operation that involves two operands namely, some text and a predefined kernel or filter, both of which

are represented as matrices containing real numbers. Typically, the matrix

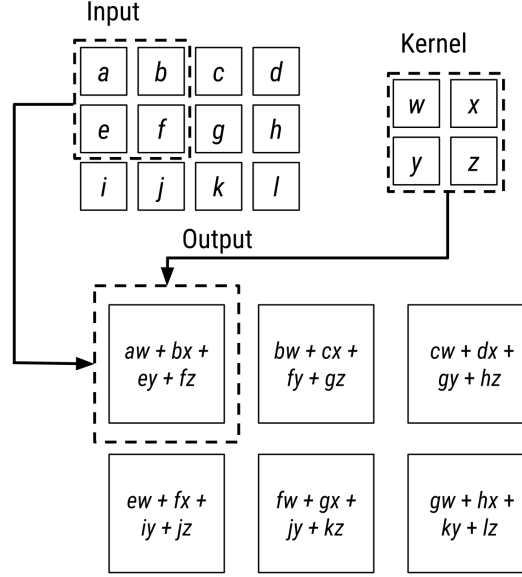


Figure 4.13: This figure illustrates the operation of convolution on a 2D image. Boxes and arrows are drawn which indicate how the convolution takes place when the kernel/filter is applied to the upper-left region of the input.

representation of text comprises of the word vectors of tokens that constitute it, that is, the word embedding matrix. The objective of the kernel is to operate on all contiguous segments of the text using a sliding window and produce real numbers of cardinality that is equal to the number of contiguous segments (in the text). This sequence of real numbers is called a feature map and is associated with a particular kernel being used to compute it. Generally, different kernels produce different feature maps that can be used as features for the task of text classification. The feature maps are then fed into a non-linear activation function as input, typically, softmax (a function that takes l real values as input and normalizes it into a probability distribution consisting of l probabilities) in our use case that outputs class or label probability estimates. The main idea here is to learn the matrix representations of kernels that in turn provide better feature maps, which optimizes the objective function efficiently. The learning takes place by predicting labels for training data and make adequate modifications to the kernel matrix through the backpropagation algorithm that minimizes the loss function, that is, minimizes the conditional log-likelihood of training data. In the aforementioned abstract elucidation, we have omitted the technical details pertaining to CNNs like mathematical definition of convolution, the objective function, and regularization. For the sake of completeness, we discuss these points in detail in the next section.

4.3.1.1.1 CNN Model Description The architecture of a CNN model with two layers including one convolution layer and a densely connected output layer is depicted in Figure 4.14. The basic component of the model is the word

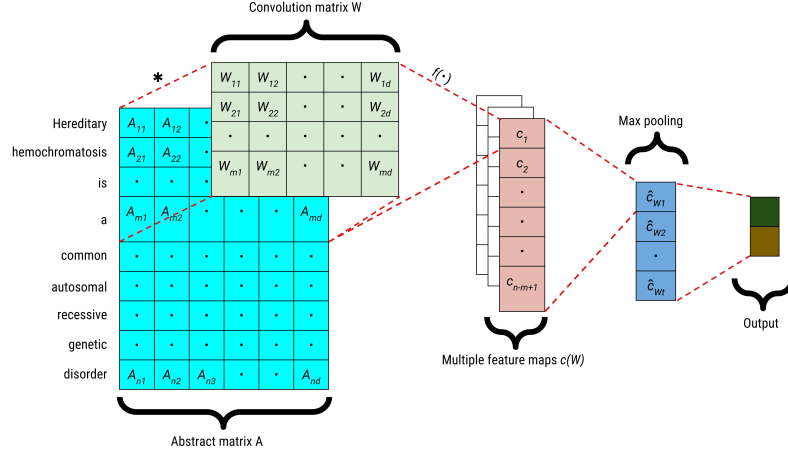


Figure 4.14: This figure illustrates a CNN with one convolution layer that is precisely used in the experimental setup in this thesis in the context of biomedical abstracts.

vector $\mathbf{x} \in \mathbb{R}^d$, where d is the dimension of the word vectors. An abstract is represented as a matrix $A \in \mathbb{R}^{n \times d}$, where n is the number of words in it. As mentioned earlier in the section 4.3.1, each row corresponds to the word vector for the corresponding token or word. For the sake of brevity, we assume that the ground truth (label) for the abstract is $y \in \mathbb{R}^2$ such that $y_2 = 1$ and $y_1 = 0$ ($y_2 = 0$ and $y_1 = 1$) when we are training on a positive (negative) instance. This precisely aligns with the Figure 4.14 with two output nodes of the final layer for the two classes (positive or negative) for each binary classifier. In this project we perform classification for 48 classes.

We define a kernel $\mathbf{W} \in \mathbb{R}^{m \times d}$, where m is the length of the sliding window. The two-dimensional convolution operation $*$ can now be defined as follows,

$$\mathbf{W} * A_{j:j+m-1} = \sum_{i=j}^{j+m-1} \sum_{k=0}^{d-1} \mathbf{W}_{i,k} A_{i,k}.$$

Next, by using a non-linear function $f(\cdot)$, we map a window of length m to a real number $c_j \in \mathbb{R}$ as

$$c_j = f(\mathbf{W} * A_{i,j:j+m-1} + b),$$

where $b \in \mathbb{R}$ is the bias. In this thesis, $f(\cdot)$ is a Rectified Linear Unit (ReLU) [17, 38]. We get the following feature map after convolving over the whole abstract,

$$\mathbf{c}(\mathbf{W}) = [c_1, \dots, c_{n-m+1}].$$

Further, to overcome the problem of abstracts of varying lengths, we perform the statistical operation of max-pooling [64] (see also Figure 4.12)

$$c_{\hat{\mathbf{W}}} = \max_i \mathbf{c}(\mathbf{W})_i,$$

that yields a feature $c_{\hat{\mathbf{W}}}$ associated to the feature map generated by \mathbf{W} . Suppose that we want to learn t kernels $\mathbf{W}^1, \dots, \mathbf{W}^t$, to obtain multiple features maps. This leads to t single max-pooled features which can be represented as follows,

$$\mathbf{c}_{\hat{\mathcal{W}}} = [c_{\hat{\mathbf{W}}^1}, \dots, c_{\hat{\mathbf{W}}^t}],$$

where $\mathcal{W} = \{\mathbf{W}^1, \dots, \mathbf{W}^t\}$. After this, finally we add a softmax layer. The parameters of the softmax layer $\mathbf{V} \in \mathbb{R}^2 \times t$ and $b^V \in \mathbb{R}^2$ with weighted inputs

$$y_j = \mathbf{V}_j \mathbf{c}_{\hat{\mathcal{W}}} + b_j^V$$

and output label probability estimates as

$$P(y_j = 1 | A, \mathcal{W}, b, \mathbf{V}, b^V) = \frac{e^{y_j}}{\sum_i e^{y_i}},$$

where \mathbf{V}_j b_j^V are the j^{th} row and j^{th} element of \mathbf{V} and b^V respectively. y_j is the j^{th} label for the abstract corresponding to matrix A .

If \mathcal{A} is the set of training abstract matrices, then in order to learn each binary classifier, we minimize the following loss/objective function

$$- \sum_{A \in \mathcal{A}} \log(P(y_j^A = 1 | A, \mathcal{W}, b, \mathbf{V}, b^V)),$$

where $j = 1$ or $j = 2$ depending upon the positive or negative training instance and y^A is the ground truth label for the abstract represented by A . The parameters of CNN, that is $(\mathcal{W}, b, \mathbf{V}, b^V)$ that minimize the loss function are modified by employing the backpropagation algorithm with stochastic gradient descent approach.

As a regularization term so that the model doesn't overfit, we use dropout [53]. More specifically, instead of feeding y_j to the softmax function during training, we actually feed

$$\hat{y}_j = \mathbf{V}_j (\mathbf{c}_{\hat{\mathcal{W}}} \circ \mathbf{s}) + b_j^V,$$

where \circ refers to hadamard product (element-wise multiplication) and $\mathbf{s} \in \{0, 1\}^t$ is generated with each element s_i drawn from the Bernoulli distribution with parameter p set to 0.2. In layman terms, this means that the gradients are backpropagated only through those neurons where $s_i = 1$.

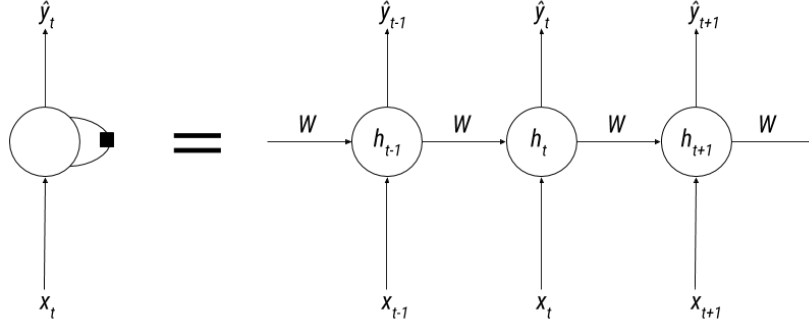


Figure 4.15: A recurrent neural network with outputs \hat{y} is depicted in the figure. The recurrent neural network processes the information/signal from the input x by incorporating the signal from unit h that is passed on forward in each time step. The figure on the left shows that block diagram of an RNN where the black square denotes a delay of a single time step. While on the right hand side, the block diagram is further unfolded and hence can be seen that each node/unit is now associated with one particular time instance.

4.3.1.2 RNN

Recurrent neural networks [50] are used for processing sequential data. Since language is sequential in nature because when we deal with natural language there is context required for predicting what comes next. RNNs are widely used in natural language models as they used to process sequential data. As a human being, every time when we try to solve something we don't reboot the brain, instead we remember things and try to build on from that. The traditional neural networks cannot have persistence they always re-learn for scratch, unlike human brain, for example, if we want to predict what word would come next in the sentence or when we want to do a prediction of the next term in the sequence. In such cases we need the information of what happened previously stored in the network, the RNNs are able to do such computations because they have loops in the network as can be seen in the Figure 4.15 [11]. They try to keep track of the information that was learnt previously. RNNs do not always perform best when handling "long term dependencies" [3]. To overcome this problem, Long Short Term Memory networks (LSTMs) were introduced by [26] that are a special kind RNNs which can handle learning the long term dependencies.

4.3.1.2.1 RNN Model Description Figure 4.15 introduces the RNN architecture where the input x_t is fed is a hidden layer at time step t . Here, each layer consists of neurons, where each neuron performs a linear matrix operation on its corresponding input which is followed by a non-linear operation. At each time step, the output of the previous step along with the next word vector in

the abstract is the input to the hidden layer that yields \hat{y} as output and features h_t . The inputs and outputs to each neuron in the RNN can be described as follows,

$$h_t = Wf(h_{t-1}) + W^{(hx)}x_t \quad (4.4)$$

$$\hat{y} = W^{(S)}f(h_t) \quad (4.5)$$

Next, we mention the details associated with each parameter in the network:

- Suppose the abstract is of n words in total. Then x_1, \dots, x_n is the word vectors corresponding to the words of the abstract.
- $h_t = \sigma(W^{hh}h_{t-1} + W^{hx}x_t)$ corresponds to the hidden layer features at a particular time step t . Here $x_t \in \mathbb{R}^d$ is the input word vector at time step t . $W^{hx} \in \mathbb{R}^{D_h \times d}$ and $W^{hh} \in \mathbb{R}^{D_h \times D_h}$ are the weight matrices used to condition input word vector x_t and output of previous time step h_{t-1} respectively.
- $\hat{y}_t = \text{softmax}(W^{(S)}h_t)$ corresponds to the output probability estimates over the vocabulary at each time step t . $W^{(S)} \in \mathbb{R}^{|V| \times D_h}$ and $\hat{y} \in \mathbb{R}^{|V|}$ where $|V|$ is vocabulary.

The RNNs are trained using an extension of the backpropagation algorithm known as backpropagation through time (BPPT) algorithm. More specifically, it begins by unfolding the RNN in time: (i) each timestep has one input, one copy of the network (every copy of the network shares the same parameters), and one output, (ii) errors are calculated and accumulated for each timestep, and (iii) the network is rolled back up and the weights are updated. Then the backpropagation algorithm is applied traditionally. The main disadvantage of RNNs is that they can not capture long-term dependencies due to vanishing/-exploding gradients during backpropagation. More specifically, if the weights of the word embedding matrix which is input to the RNN are small, it can lead to vanishing gradients where the gradient signal is so weak that the learning is slow or stops altogether. Conversely, if the weights are large, it can lead to very strong gradient signal that causes the learning to diverge, that is, exploding gradients.

To overcome the problem of long-term dependencies, LSTM was developed, that is, to gain more persistent memory. The mathematical formulation of the LSTM unit is described below,

$$\begin{aligned}
i^{(t)} &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) && \text{(Input Gate)} \\
f^{(t)} &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) && \text{(Forget Gate)} \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) && \text{(Output Gate)} \\
\tilde{c}^{(t)} &= \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) && \text{(New Memory cell)} \\
c^{(t)} &= f^{(t)} \circ \tilde{c}^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{(Final memory cell)} \\
h^{(t)} &= o_t \circ \tanh(c^{(t)}) &&
\end{aligned} \tag{4.6}$$

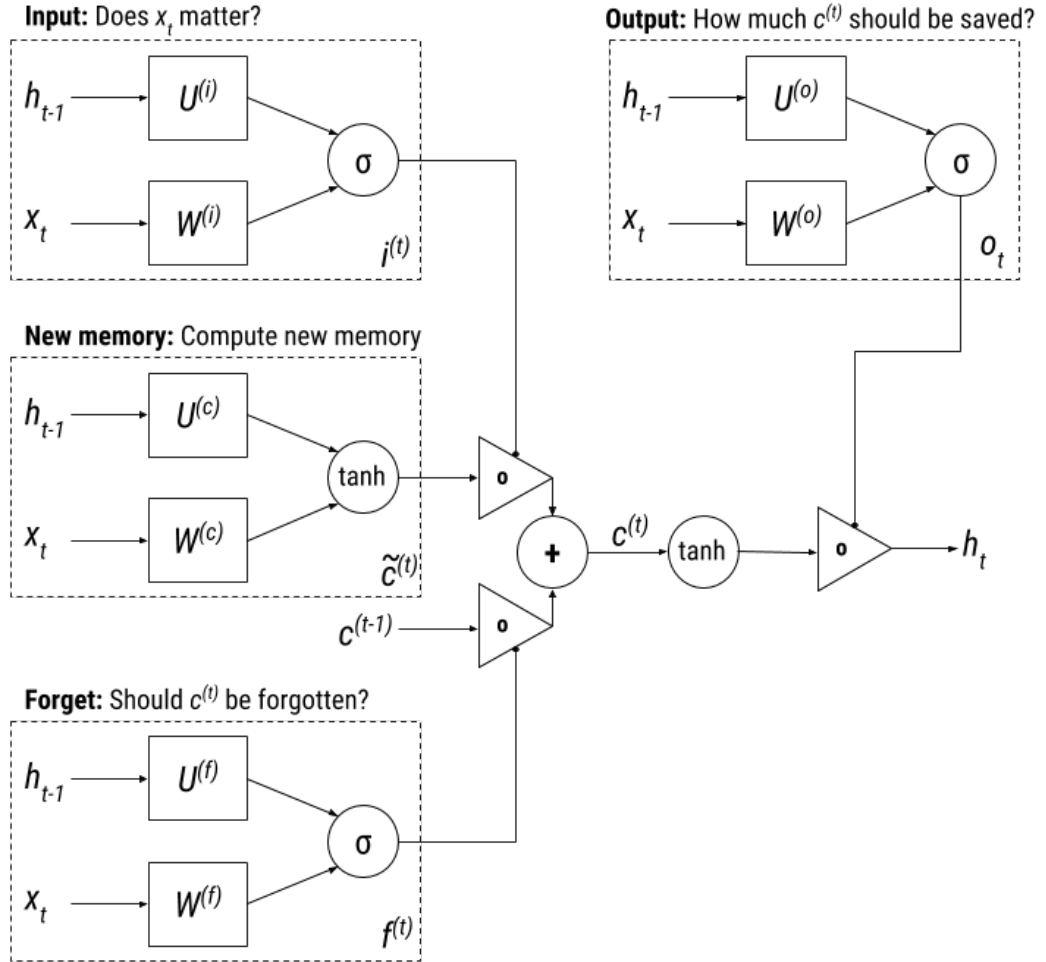


Figure 4.16: The detailed internals of a LSTM unit

Finally, we explain the structure and the intuition behind LSTM units below:

- **New memory generation:** It is the consolidation of a new input word vector x_t with the past hidden state h_{t-1} . From the model point of view, this gate cooks the recipe of summarizing the new input in the light of contextual past.
- **Input Gate:** The new memory cell generates new memory blindly without even bothering whether the new input is important or not – the objective of the input gate is to determine the importance to the new input by using it and the past hidden state. Hence, it is used to gate the new memory and produces $i^{(t)}$ which indicates information.
- **Forget Gate:** The forget gate is analogous to input gate in the sense that it makes an assessment on the usefulness of the past memory cell that is used for the computation of current memory cell. Thus, it looks at the input and the past hidden state and produces $f^{(t)}$.
- **Final memory generation:** Both the input gate and the forget gate act as an advisory to final memory cell, that is, whether to forget past memory or gate new memory respectively. The final memory cell then sums these results to produce the final memory $c^{(t)}$.
- **Output Gate:** Its objective is to separate final memory $c^{(t)}$ from the hidden state. $c^{(t)}$ might not necessarily contain only relevant information all the time that is required to be saved by the hidden state. Moreover, as hidden states are used in every gate of an LSTM unit, the output gate makes an assessment regarding the importance of the information to be stored in the hidden state h_t . This produces o_t and this is to gate the point-wise \tanh of the memory.

The bi-directional version of RNN or LSTM, at each time-step t , maintains two hidden layers, one for left-to-right propagation and another for right-to-left propagation. Clearly, this consumes two times as much memory as RNNs. The final output \hat{y}_t is obtained by combining the scores computed by both the hidden states.

Chapter 5

Results and Discussion

In this thesis, the research problem that we are solving is to automatically predict experimental metadata using scientific publications. For such a problem we dive into the realm of text classification because we use scientific publications (unstructured text). As mentioned in Chapter 4 for classification in the text a set of labels/categories are required. Therefore, for this purpose, we used the NCBI disease corpus (explained in Section 3.2) which contains a set of publications which has annotated disease mentions. Here, we only predict this metadata category disease, as it is a relevant term in the biomedical domain.

We first start this chapter by defining the evaluation metrics that have been presented in the results. Firstly these metrics are defined for binary classification and then using an example the metrics are defined for multi-label classification.

Performance measures

For evaluating results from classification, we need measures to assess the performance of the method. Hence, for this purpose statistical measures are used based on the confusion matrix shown in Table 5.1. In the confusion matrix, TP is the number of times the positive class (1) is classified correctly, FP means the number of times the negative class (0) is misclassified as positive (1), TN means the number of times the negative class is classified correctly and FN is the number of times the positive class (1) is classified incorrectly.

		Predicted	
		0	1
Actual	0	true negative (TN)	false positive (FP)
	1	false negative (FN)	true positive (TP)

Table 5.1: Confusion Matrix - table that is used to calculate evaluation metrics

The metrics that have been used in this project are defined as follows:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$, which is the percentage of times where the model gives correct results.
- Precision = $\frac{TP}{TP+FP}$, this is also known as the positive predictive value. This metric tells the percentage of relevant results among the predicted results.
- Recall = $\frac{TP}{TP+FN}$, also known as sensitivity. This is the percentage of positive instances classified correctly among all the correctly classified instances.
- F1 score = $\frac{2*precision*recall}{precision+recall}$, this is used when we have a class which has a smaller number of occurrences. It helps in combining the trade-offs of precision and recall.

Now we define the metrics in terms of multi-label classification. For this purpose we use an example demonstrated in Table 5.2:

Input	$y^{(i)}$ (Actual label)	$\hat{y}^{(i)}$ (Predicted Labels)
$\tilde{x}^{(1)}$	[1 0 1 0]	[1 0 0 1]
$\tilde{x}^{(2)}$	[0 1 0 1]	[0 1 0 1]
$\tilde{x}^{(3)}$	[1 0 0 1]	[1 0 0 1]
$\tilde{x}^{(4)}$	[0 1 1 0]	[0 1 0 0]
$\tilde{x}^{(5)}$	[1 0 0 0]	[1 0 0 1]

Table 5.2: An example of predictions in a multi-label classification to depict evaluation metrics

We start by defining the accuracy:

$$Accuracy = \frac{1}{N} \sum_{i=1}^N \frac{|\hat{y}^{(i)} \wedge y^i|}{|\hat{y}^{(i)} \vee y^i|} \quad (5.1)$$

where \wedge and \vee are logical OR and AND operations, which are applied vector-wise. Then we define the F_1 measure for the multi-label classification:

$$F_1 = \frac{1}{N} \sum_{i=1}^N \frac{2|\hat{y}^{(i)} \wedge y^i|}{|\hat{y}^{(i)}| + |y^i|} \quad (5.2)$$

Overall, higher the value of accuracy and F_1 score the better the learning algorithm.

We start by presenting a baseline approach to this research problem. Then we move on to the comparison of results (judging mainly by the accuracy). The results are presented using the neural network model and they are compared with baseline multi-label classification methods. The comparison is divided into two parts (i) predicting the disease terms and (ii) predicting the superclass of the disease terms.

The dataset was tested with three types of problem transformations (see Section 4.2): (i) Binary Relevance (BR), (ii) Label Powerset (LP) and (iii) Classifier Chains (CC). The problem transformations were used from scikit-multilearn that is a library specific to multi-label classification built on top of the well-known scikit-learn. [52]. The classifiers that were tested using these transformations were

- Multi-layer Perceptron classifier (MLP classifier)
- Multi-label k Nearest Neighbour (MLkNN)
- Random Forest Classifier
- Decision Trees Classifier

For each combination the evaluation metrics used were: accuracy and weighted F1 score. The classifiers used and the metrics were calculated using scikit-learn [44].

For the neural network approach, we utilized the pretrained word embeddings - GloVe: Global Vectors for Word Representation [45] of dimension 300. For data preprocessing we used tokenizer API from Keras [10]. The CNN and BiLSTM architectures were used from PyTorch¹. For validating the results using the neural networks we split the data into training and validation sets. The training set is 80% of the whole data and the validation set is the rest 20%. A similar train-test split was done for the baseline multi-label classification 80%-20%. The dropout rate has been kept 0.2 in all the neural network models.

5.1 A baseline approach

In the prediction of metadata, specifically diseases - a baseline approach would be to use a vocabulary of disease names and perform a basic search in the documents. If we have a match it can be said that a particular publication talks about ‘xyz’ disease. This would be a direct and trivial way to approach this problem.

In this case, the disease names from MeSH were downloaded². After that, direct matching is done using Flashtext [52]. This also includes the synonyms so that the matching is not biased for terms like ‘heart disease’ and ‘cardiovascular disease’. For example in the abstract text heart disease is mentioned but in the list of the downloaded disease terms, cardiovascular diseases are present. Even after including the synonyms we could find only 44 unique terms in the text, which is 7.5% of the identified disease terms (588).

¹<https://pytorch.org/>

²<http://bioportal.bioontology.org/ontologies/MESH/?p=classes&conceptid=root>

5.2 Predicting Disease Terms

For prediction of disease terms, the unique IDs (explained in Section 3.2) are used as labels after preprocessing - removing duplicates and characters if present.

5.2.1 Multi-label Classification

For baseline multi-label classification two types of feature extraction techniques were used: (i) TF-IDF and (ii) Bag of words representation (BoW). The results for these two techniques are presented in Tables 5.3 and 5.4. For both the techniques the maximum feature length is kept at 200 (reason explain in Section 5.4). It can be seen that the MLP Classifier performs the best in both the techniques of feature extraction.

There is a difference in the performance if we compare the different feature extraction methods, the TF-IDF technique performs better than BoW. The TF-IDF normalizes the outputs which is why this technique is known to perform better when comes to the feature selection, it can be observed from the results as well. Since dataset has a large number of diseases - 588, and a limited number of abstracts, the accuracy of predictions is not very high. The highest accuracy that we get here is $\approx 19\%$.

Classifiers	Random forest			Decision Tree			MLPClassifier			MLkNN			
Problem Transformations	BR	LP	CC	BR	LP	CC	BR	LP	CC	k=20	k=30	k=10	k=5
Accuracy	0.101	0.075	0.088	0.044	0.107	0.082	0.170	0.189	0.164	0.164	0.138	0.176	0.176
F1 score	0.125	0.191	0.117	0.259	0.208	0.250	0.207	0.233	0.203	0.194	0.184	0.238	0.232

Table 5.3: Results for Disease terms using TF-IDF feature selection

Classifiers	Random forest			Decision Tree			MLPClassifier			MLkNN			
Problem transformations	BR	LP	CC	BR	LP	CC	BR	LP	CC	k=20	k=30	k=10	k=5
Accuracy	0.088	0.126	0.088	0.082	0.094	0.113	0.132	0.157	0.126	0.088	0.088	0.107	0.082
F1 score	0.106	0.160	0.090	0.256	0.191	0.257	0.175	0.217	0.171	0.098	0.091	0.128	0.163

Table 5.4: Results for Disease terms using Bag of words feature selection

5.2.2 Neural Network Methods

For the neural network method, two models were trained - (i) BiLSTM and (ii) CNN. The embedding dimension in both cases is 300, where the input sequences are kept as 200 for which the results are discussed. The models are also retrained for a sequence length of 150 and 250 to check whether the accuracy increases or decreases. The models are trained using the Adam optimizer [32]. The results of the models are presented in Table 5.5. The number of epochs is kept high because of a large number of labels.

From Table 5.6, it can be seen that the CNN outperforms all the baseline methods and the BiLSTM. The BiLSTM has a lesser accuracy than the highest

	Validation accuracy	Sequence Length
BiLSTM	0.177	200
	0.168	150
	0.099	250
CNN	0.31	200
	0.283	150
	0.335	250

Table 5.5: Results for deep neural networks for predicting disease names

baseline MLP classifier. One explanation would be that since the RNN treats the input as a sequence, and its a very long sequence, as it goes ahead working on word after word, it forgets what happened in the words before. Even though we’re using a bidirectional RNN, maybe the left to right forgets what happened at the start, say by the time it reaches mid sequence, and the right to left forgets what it saw in the first (rightmost) terms by the time it reaches mid sequence too.

Whereas in CNNs, the convolution kernels are time invariant so they cannot distinguish between different parts of the sequence. As a disadvantage, they cannot easily do inferences which require using context and need to treat the input as a sequence. However, in this case, it might not be needed. Abstracts may somewhere mention a particular disease keyword which is enough on its own to detect which class it belongs to (and this detection requires no sequential treatment of input, which is similar to ‘find a word’).

Method	Accuracy
MLP Classifier (TF-IDF)	0.189
MLP Classifier (BoW)	0.157
CNN (length = 250)	0.335
BiLSTM (length = 200)	0.177

Table 5.6: Best performing methods when predicting disease terms

Moreover, we see from Table 5.5 that when the sequence length increases to 250 the accuracy for CNN increase, this can be because the disease terms would be occurring towards the end of the abstract. For further checking this hypothesis we perform experiments when we only consider some part of the abstract (see Table 5.7). For this, the abstracts are split into sentences and only a subset of these sentences are used per abstract.

From Table 5.7 we see that the accuracy does not increase when we use the last two sentences as we mentioned above. Therefore, such a generalization is

	Validation Accuracy	Sentences of abstract used
BiLSTM	0.074	First two
	0.073	Last two
	0.103	First and last
CNN	0.268	First two
	0.133	Last two
	0.28	First and last

Table 5.7: Results for predicting disease terms using deep neural nets when only some part of abstract is taken as input

not correct. Another interesting observation, in this case, is that both BiLSTM and CNN have higher accuracy when the first and last sentence are present in the abstract. This would be due to the fact that human beings read and write the first and last sentence very carefully with a lot of information which aligns to the myth about how humans read and write. Moreover, it adheres to how humans do fast reading.

5.3 Predicting Super Classes

In this experiment we use the MeSH tree structure³ to map the annotated disease names to their superclasses (or parent classes) of the disease terms as shown in Figure 5.1. Since the dataset we use involves the disease terms from the Medical Subject Headings (MeSH) ontology - which provides hierarchically-organized terminology for indexing diseases. For example the disease term “Hemochromatosis” belongs to the superclass ‘Nutritional and Metabolic Diseases [C18]’ and ‘Congenital, Hereditary, and Neonatal Diseases and Abnormalities [C16]’. To do so we use the unique ID of the disease terms as explained in Section 3.2 to perform a SPARQL query to map these terms (refer Listing 5.1).

Originally we want to map the disease terms to the **26** superclasses as depicted in Figure 5.1, however, the query returns 48 classes. The additional 24 additional classes that are returned as a result of the query (Listing 5.1) are mentioned below:

‘Physical Phenomena’, ‘Genetic Phenomena’, ‘Population Characteristics’, ‘Nonsyndromic sensorineural hearing loss’, ‘Diagnosis’, ‘Physiological Phenomena’, ‘Psychological Phenomena’, ‘Cell Physiological Phenomena’, ‘Investigative Techniques’, ‘Biological Phenomena’, ‘Behavior and Behavior Mechanisms’, ‘Immune System Phenomena’, ‘Reproductive and Urinary Physiological Phenomena’, ‘Fluids and Secretions’, ‘Cells’, ‘Health Occupations’, ‘Environment and Public Health’, ‘Tissues’, ‘Mental Disorders’, ‘Behavioral Disciplines and Activities’, ‘Musculoskeletal and Neural Physiological Phenomena’, ‘Therapeutics’, ‘Health Care Quality, Access, and Evaluation’, ‘Natural Science Disciplines’.

³<https://meshb.nlm.nih.gov/treeView>

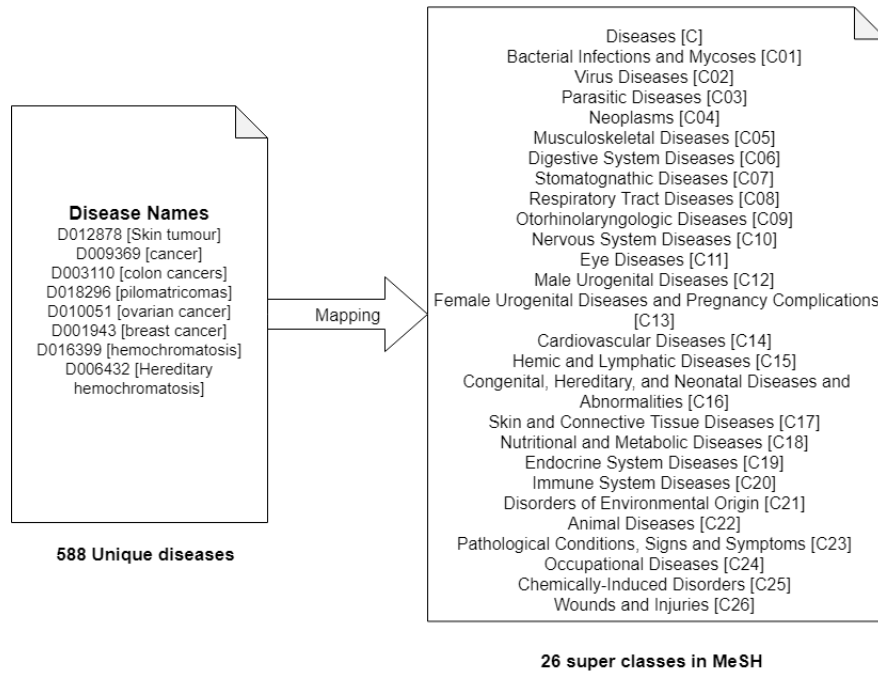


Figure 5.1: Mapping superclasses in MeSH

This happens due to the fact that one disease term is sometimes present under more than one parent class. For example the terms ‘Genetic Phenomena (ID: [G05])’ and ‘Pathological Conditions, Signs and Symptoms [C23]’ are superclasses itself for the disease name ‘Chromosome Aberrations’ as can be seen in <https://meshb.nlm.nih.gov/record/ui?ui=D002869>.

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
4 PREFIX owl: <http://www.w3.org/2002/07/owl#>
5 PREFIX meshv: <http://id.nlm.nih.gov/mesh/vocab#>
6 PREFIX mesh: <http://id.nlm.nih.gov/mesh/>
7 PREFIX mesh2015: <http://id.nlm.nih.gov/mesh/2015/>
8 PREFIX mesh2016: <http://id.nlm.nih.gov/mesh/2016/>
9 PREFIX mesh2017: <http://id.nlm.nih.gov/mesh/2017/>
10 PREFIX mesh2018: <http://id.nlm.nih.gov/mesh/2018/>
11 PREFIX mesh2019: <http://id.nlm.nih.gov/mesh/2019/>
12 SELECT DISTINCT ?p ?label ?uri
13 WHERE {
14   mesh:D012878 meshv:broaderDescriptor* ?uri .
15   ?uri rdfs:label ?p.
16   FILTER NOT EXISTS{
17     ?uri meshv:broaderDescriptor ?x
18   }
19 }
```

Listing 5.1: SPARQL query to map *one* disease name to the respective super class using the MeSH SPARQL endpoint (<https://id.nlm.nih.gov/mesh/query>)

As a result of extracting the super classes for disease terms, the number of labels is reduced to 48. In this experiment we perform text classification for these 48 labels.

5.3.1 Multi-label Classification

The results for the baselines are present in Table 5.8, here the features were extracted using TF-IDF. Another set of features were extracted using the bag of word representation, the results are shown in the Table 5.9. The maximum number of features for both the feature extraction techniques is kept to be 200. The Label Powerset transformation gives better performance for all three baseline classifiers. Overall, the Random forest classifier has the highest accuracy with problem transformation of Label Powerset.

In this setup, the feature extraction using the bag of words technique has a better performance when compared to TF-IDF. An explanation behind this would be that in a random forest classification method when given a set of features and labels it, creates random subsets of features. Then building decision trees with the help of these subsets after which it makes predictions.

Classifier Problem Transformation	Random forest			Decision Tree			MLPClassifier			MLkNN			
	BR	LP	CC	BR	LP	CC	BR	LP	CC	k=20	k=30	k=10	k=5
Accuracy	0.181	0.348	0.174	0.148	0.271	0.135	0.232	0.348	0.335	0.303	0.284	0.316	0.297
F1 score	0.605	0.642	0.572	0.676	0.632	0.662	0.669	0.712	0.668	0.670	0.669	0.681	0.683

Table 5.8: Results of predicting super class using TF-IDF feature selection

	Random forest			Decision Tree			MLPClassifier			MLkNN			
	BR	LP	CC	BR	LP	CC	BR	LP	CC	k=20	k=30	k=10	k=5
Accuracy	0.168	0.368	0.200	0.116	0.329	0.168	0.258	0.342	0.303	0.148	0.097	0.142	0.142
F1 score	0.605	0.649	0.554	0.641	0.657	0.630	0.668	0.672	0.674	0.520	0.510	0.521	0.531

Table 5.9: Results of predicting super class using Bag of words feature selection

5.3.2 Neural Network Methods

For the neural network method, two models were trained - (i) BiLSTM and (ii) CNN. The dimension of the embedding matrix in both cases is 300, where the input sequences of the abstract are kept as 200 for which the results are discussed. The models are also retrained for a sequence length of 150 and 250 to check whether the accuracy increases or decreases. The models are trained using the Adam optimizer [32]. The results are shown in the Table 5.10. The number of epochs is kept to 65 as the number of labels is not high as in the previous setup.

Here both the BiLSTM and CNN outperform all the baseline classifiers. Overall, CNN has the highest performance accuracy with the sequence length of 200. Another interesting observation is that when the sequence length is

	Validation Accuracy	Sequence Length
BiLSTM	0.623	200
	0.628	150
	0.697	250
CNN	0.837	200
	0.76	150
	0.803	250

Table 5.10: Results for deep neural networks for predicting super class

increased to 250 the accuracy of the BiLSTM increases, whereas one would think it should decrease because LSTMs tend to forget what happened previously or ahead when the sequence length is increased. One explanation behind this could be that the disease words occur towards the end in the abstract. When we use the last two sentences per abstract for predicting the superclass in BiLSTM the accuracy decreases to 48.4% (see Table 5.12). Therefore, a generalization of such kind cannot be made. Similar to predicting disease terms here also the model performs better when the first and last sentence is taken as the input.

Method	Accuracy
Random Forest and MLP Classifier (TF-IDF)	0.348
Random Forest (BoW)	0.368
CNN (length = 200)	0.837
BiLSTM (length = 250)	0.697

Table 5.11: Best performance among the different methods when predicting super classes

	Validation Accuracy	Sentences of abstract used
BiLSTM	0.518	First two
	0.484	Last two
	0.60	First and last
CNN	0.76	First two
	0.771	Last two
	0.787	First and last

Table 5.12: Results for predicting super classes using deep neural nets when only some part of abstract is taken as input

5.4 Discussion

In this section, we mainly discuss the results that we obtain using the neural network method, specifically for the sequence length of 200. The reason why we keep this as a gold standard input length is that it is closest to the mean length of the number of words in all the abstracts (refer Section 3.2). Furthermore, we also observed from the results that by increasing the sequence length the performance does not significantly increase. Therefore, rather than training at a maximum length we use the mean length as it also saves the running time of the experiments.

For this discussion, we pick three abstracts that are common in the validation set of both the disease term and superclass experiment. Note that the text is preprocessed.

Abstract 1: promoter luciferase constructs were transiently cotransfected with a wild type vhl wt vhl vector in several cell lines including 293 embryonic kidney and rcc cell lines wt vhl protein inhibited vegf promoter activity in a dose dependent manner up to 5 to 10 fold deletion analysis defined a 144 bp region of the vegf promoter necessary for vhl repression this vhl responsive element is gc rich and specifically binds the transcription factor sp1 in crude nuclear extracts in drosophila cells cotransfected vhl represses sp1 mediated activation but not basal activity of the vegf promoter we next demonstrated in coimmunoprecipitates that vhl and sp1 were part of the same complex and by using a glutathione s transferase vhl fusion protein and purified sp1 that vhl and sp1 directly interact furthermore endogenous vegf mrna levels were suppressed in permanent rcc cell lines expressing wt vhl and nuclear run on studies indicated that vhl regulation of vegf occurs at least partly at the transcriptional level these observations support a new mechanism for vhl mediated transcriptional repression via a direct inhibitory action on sp1 and suggest that loss of sp1 inhibition may be important in the pathogenesis of von hippel lindau disease and rcc.

Actual labels:

Superclass - ['Congenital, Hereditary, and Neonatal Diseases and Abnormalities', 'Nervous System Diseases', 'Cardiovascular Diseases', 'Male Urogenital Diseases', 'Female Urogenital Diseases and Pregnancy Complications', 'Skin and Connective Tissue Diseases', 'Neoplasms']

Disease Names - ['von Hippel-Lindau Disease', 'Carcinoma, Renal Cell'].

Abstract 2: we have analyzed the 27 exons and the promoter region of the *rb1* gene in familial or sporadic bilateral retinoblastoma by using single strand conformation polymorphism analysis for improvement over previous studies a new set of primers has been designed which allow for amplification of the coding and splicing sequences only the positioning of the polymerase chain reaction pcr primers was such that the resulting pcr products were of different sizes which enabled us to analyze two different exons simultaneously and still distinguish between the banding profiles for both biplex analysis by using this approach we were able to identify mutation in 22 new patients but the overall efficiency of the procedure when we used a single pass regimen was only 48 the mutations were small insertions and deletions and point mutations in roughly equal proportions

Actual labels: *Super class* - ['Congenital, Hereditary, and Neonatal Diseases and Abnormalities', 'Neoplasms', 'Eye Diseases']

Disease names - ['Retinoblastoma']

Abstract 3: programmed cell death or apoptosis is a physiological process essential to the normal development and homeostatic maintenance of the immune system the fas apo 1 receptor plays a crucial role in the regulation of apoptosis as demonstrated by lymphoproliferation in *mrl lpr lpr* mice and by the recently described autoimmune lymphoproliferative syndrome *alps* in humans both of which are due to mutations in the *fas* gene we describe a novel family with *alps* in which three affected siblings carry two distinct missense mutations on both the *fas* gene alleles and show lack of fas induced apoptosis the children share common clinical features including splenomegaly and lymphadenopathy but only one developed severe autoimmune manifestations in all three siblings we demonstrated the presence of anergic cd3 cd4 cd8 double negative dn t cells moreover a chronic lymphocyte activation was found as demonstrated by the presence of high levels of hla dr expression on peripheral cd3 cells and by the presence of high levels of serum activation markers such as soluble interleukin 2 receptor *sll 2r* and soluble cd30 *scd30*

Actual Label: *Super class* - ['Congenital, Hereditary, and Neonatal Diseases and Abnormalities', 'Pathological Conditions, Signs and Symptoms', 'Immune System Diseases', 'Hemic and Lymphatic Diseases']

Disease Names - ['Autoimmune Lymphoproliferative Syndrome', 'Splenomegaly', 'Lymphatic Diseases', 'Autoimmune Diseases']

The predictions of the three abstracts are presented in Tables 5.13 and 5.14. From the predictions, it can be seen when we use a large number of labels(588), in $\approx 38\%$ abstracts in the validation set there is no prediction at all for both

	CNN	
	Predicted Disease Names	Predicted Super Classes
Abstract 1		‘Cardiovascular Diseases’, ‘Female Urogenital Diseases and Pregnancy Complications’, ‘Nervous System Diseases’, ‘Skin and Connective Tissue Diseases’, ‘Nutritional and Metabolic Diseases’, ‘Neoplasms’, ‘Congenital, Hereditary, and Neonatal Diseases and Abnormalities’
Abstract 2	‘Retinoblastoma’	‘Congenital, Hereditary, and Neonatal Diseases and Abnormalities’, ‘Neoplasms’, ‘Eye Diseases’
Abstract 3		‘Congenital, Hereditary, and Neonatal Diseases and Abnormalities’

Table 5.13: Prediction by CNN for the three abstracts

	BiLSTM	
	Predicted Disease Name	Predicted Super Class
Abstract 1	‘Neoplasms’	‘Cardiovascular Diseases’, ‘Neoplasms’, ‘Skin and Connective Tissue Diseases’
Abstract 2		‘Neoplasms’, ‘Digestive System Diseases’, ‘Congenital, Hereditary, and Neonatal Diseases and Abnormalities’, ‘Skin and Connective Tissue Diseases’
Abstract 3		‘Immune System Diseases’, ‘Hemic and Lymphatic Diseases’, ‘Congenital, Hereditary, and Neonatal Diseases and Abnormalities’, ‘Cardiovascular Diseases’, ‘Nutritional and Metabolic Diseases’

Table 5.14: Prediction by BiLSTM for the three abstracts

CNN and BiLSTM. For this, we need a bigger dataset for the models to train on. If we get a larger annotated set, the model will be able to learn in a much better way as it would have seen a good amount of examples for such a large set of labels.

We observed that when predicting super classes CNN is very accurate(83%) as it would have seen enough examples to learn which label would fit where. Similarly, BiLSTM is accurate more than 50% of times overall even in some cases where CNN isn’t, for example in Abstract 3. This could be because in this case the context of the language would be required more rather than in the other cases. For example, the abstract 3 talks about immune system diseases so in this case, the BiLSTM learns this whereas CNN would not be able to do that. In some cases, no result comes because the last layer is the softmax layer and the probability of each category would be less than 0.5, so the model does not output any disease.

For evaluating whether there is any bias present in the predictions for super-classes, we perform an analysis of the frequency of the labels. First we compute the frequency of each label for the whole dataset, then specifically for the valida-

tion set. This is done for both the CNN and BiLSTM. From the analysis, it was observed that the superclass “Congenital, Hereditary, and Neonatal Diseases and Abnormalities” has the highest frequency of occurrence (639). Therefore, even in the validation set for CNN and for BiLSTM both, it has the highest frequency. Note that in the validation set, the frequency is calculated separately for predictions and the actual labels. In the predictions, the frequency was 139 whereas in the actual labels it occurs 125 times for BiLSTM. Similarly, for CNN the frequency in actual labels was 144 whereas the frequency is 126 in the predicted labels. From the frequency, we observe that the model predicts better if the label has a higher number of occurrences.

We perform a similar analysis for the less frequently occurring labels: “Bacterial Infections and Mycoses”. This label has a frequency of 23 in the entire label set. After looking into the validation set for both CNN and BiLSTM, we observed that in CNN it is predicted 7 times but actually it occurs only once. For BiLSTM, it is predicted only once, but in actual labels, it has a frequency of 7. Since the overall occurrence of “Bacterial Infections and Mycoses” is not higher the predictions are not as accurate as we observe before with the label “Congenital, Hereditary, and Neonatal Diseases and Abnormalities”.

From this analysis, we see that a large set of disease names has one particular, therefore, there is a bias when we try to predict these superclasses. Moreover, the input corpus contains 793 abstracts in total, which only gives a particular subset of diseases. Hence there is a bias in prediction related to the frequency of occurrences when extracting the superclasses. From the results, we saw that CNN gives promising results in both experimental setups – predicting disease terms and predicting superclasses. If we want to involve more metadata categories like tissue, cell line etc, it would be easily incorporated in the model as we train a multi-label classification problem.

Chapter 6

Conclusions and Future Work

In this thesis project, we focus on the research problem of automatically predicting experimental metadata using scientific publications. This is done to tackle the problem of poor quality metadata. To make data reusable we need to assess the quality of metadata, and we hypothesize that using scientific publications for prediction of experimental metadata would help with increasing this quality. We also formalize five research questions toward answering this vast research problem.

In chapter 2 we discussed the literature survey - related work done close to this research problem. In chapter 3 provided a brief description of the methods used and a description of the dataset. In chapter 4 we discussed the model specification of CNN and BiLSTM in the context of NLP. Lastly in Chapter 5 we presented an analysis of the results. In this chapter 6 we discuss the conclusion of this work (Section 6.1), the limitations and future work are listed in Section 6.2.

6.1 Conclusions

We started by looking for annotated datasets, which use standard vocabulary so that we can train a model with that dataset. We started with the metadata key “disease”, as we found a corpus - NCBI disease corpus (refer Section 3.2). We trained this data on mainly used two architectures - CNN and BiLSTM. We perform two different kinds of experiments with these two architectures (i) first we predict the disease names by using their unique ID in the MeSH ontology and (ii) second, we use the tree structure of MeSH ontology to move up in the hierarchy of these disease terms which reduces the number of labels. We also perform various multi-label classification techniques for the above mentioned two experiments. We used two different feature extraction techniques for these baseline classifiers - (i) TF-IDF and (ii) Bag of Words. Additionally, we also perform a baseline method of direct matching the disease terms in the text. We

select the metadata key *disease* for prediction as it is very essential and relevant in the biomedical domain.

We used the NCBI disease corpus, to predict the disease names using 793 PubMed abstracts. We used GLoVe word embedding to train the CNN and BiLSTM. Overall, the CNN model performed the best among all the methods that were used for prediction. Now we answer all the research questions that were outlined in Chapter 1.

- RQ1: Up to what extent can we predict metadata from scientific publications?
- RQ2: Which specific metadata fields can be accurately be predicted?
- RQ3: What are the optimal set of features that should be selected for the machine learning algorithm to accurately predict metadata fields or the prediction is better without employing features at all?
- RQ4: Which is the best automated (machine learning/deep learning) method to predict metadata?
- RQ5: To what extent does the accuracy improves when we use abstract and full text?

6.1.1 Research Question 1 & 2

In this project, we only look at the prediction of metadata key type ‘disease’. We performed two different types of experiments (i) predicting disease names and (ii) predicting superclasses of disease names (refer Chapter 5). In the first experiment we found that CNN performed best with $\approx 30\%$ accuracy, but when it comes to predicting the superclasses CNN gave a very high accuracy of $\approx 80\%$. This is a promising result, which can be employed when we try to predict more than one metadata key type.

Since we used a limited number (793) of abstracts therefore, to predict the disease names we did not get a high accuracy in the first experiment. For predicting a large number of labels we need a higher number of abstracts for the accuracy to improve.

6.1.2 Research Question 3 & 4

We used baseline multi-label classification with two different feature extraction techniques - a bag of words and TF-IDF. We saw that the TF-IDF feature extraction technique worked better when predicting the disease terms, but the bag of words technique worked better when predicting superclasses (refer Chapter 5).

From the results, we saw that rather than following the traditional methods of extracting features in the dataset and then select a machine learning model, if we use a pre-trained word embedding we get better results. In this case, we used GloVe: Global Vectors for Word Representation [45]. We use a pre-trained word

embedding because it contains global information about a word and training a word embedding from scratch requires a large amount of data.

6.1.3 Research Question 5

In this project, we did not include the full-text articles when performing experiments. In the biomedical field, a lot of articles hide behind paycheck walls. One would need to filter out the full access articles which would be very time consuming.

In conclusion, we saw that traditional methods of performing feature engineering do not perform best when it comes to natural language processing. We would need a numerous number of rules to cover all the ambiguities and exceptions in the language. Additionally, we would always need a different set of rules when dealing with data from different domains, which wouldn't be a trivial task. These methods require more time, cost and a lot of manual effort.

On the contrary, when we use deep learning we don't have to provide feature specifically, the model learns the features on its own. Therefore, using pre-trained word embeddings give the idea of a language model to the network, this is where it also learns context and ambiguities. Hence when we want to predict various metadata keys in the future, a deep learning model would be a promising technique to move further.

6.2 Limitations and Future Work

A major limitation to this problem is the availability of a large gold standard dataset on which a model could be trained. We need a large annotated dataset which could be used to train a model which could then be used to predict metadata keys and values. The corpus that was used for this project had a limited amount of abstracts with only disease mentions that were annotated - which is the reason the poor results for predicting disease terms.

As a part of future work, we can extend our model to predict for other metadata key types like an organism, tissue, cell line etc. A more context specific word embedding could be used in the deep learning architecture that we used to check whether this would help improve the predictions of the metadata. Transfer learning could be used - "Transfer learning and domain adaptation refer to the situation where what has been learned in one setting - is exploited to improve generalization in another setting [19]". Moreover, in order to use transfer learning, we train the model on a large annotated corpus like the data present in BioASQ challenge¹ and test in on the NCBI disease corpus.

¹<http://bioasq.org/participate/challenges>

Bibliography

- [1] Tanya Barrett, Stephen E Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F Kim, Maxim Tomashevsky, Kimberly A Marshall, Katherine H Phillippy, Patti M Sherman, Michelle Holko, et al. Ncbi geo: archive for functional genomics data sets—update. *Nucleic acids research*, 41(D1):D991–D995, 2012.
- [2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.
- [3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- [4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
- [5] Christine L Borgman. The conundrum of sharing research data. *Journal of the American Society for Information Science and Technology*, 63(6):1059–1078, 2012.
- [6] Alvis Brazma, Pascal Hingamp, John Quackenbush, Gavin Sherlock, Paul Spellman, Chris Stoeckert, John Aach, Wilhelm Ansorge, Catherine A Ball, Helen C Causton, et al. Minimum information about a microarray experiment (miame)—toward standards for microarray data. *Nature genetics*, 29(4):365, 2001.
- [7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [8] Leo Breiman. *Classification and regression trees*. Routledge, 2017.
- [9] Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. How to train good word embeddings for biomedical nlp. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174, 2016.
- [10] François Chollet et al. Keras. <https://keras.io>, 2015.

- [11] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.
- [12] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- [13] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, volume 10, pages 279–286, 2010.
- [14] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10, 2014.
- [15] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.
- [16] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.
- [17] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.
- [18] Rafael S Gonçalves, Martin J O’Connor, Marcos Martínez-Romero, John Graybeal, and Mark A Musen. Metadata in the BioSample Online Repository are Impaired by Numerous Anomalies. *arXiv preprint arXiv:1708.01286*, 2017.
- [19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [20] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.
- [21] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995.
- [22] Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48, 2017.

- [23] Steve Harris, Andy Seaborne, and Eric Prud’hommeaux. Sparql 1.1 query language. *W3C recommendation*, 21(10), 2013.
- [24] Simon Haykin. *Neural networks*, volume 2. Prentice hall New York, 1994.
- [25] Jeff Heflin. Owl web ontology language use cases and requirements. <http://www.w3.org/TR/webont-req/>, 2004.
- [26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [27] Sharona Hoffman and Andy Podgurski. The use and misuse of biomedical data: is bigger really better? *American journal of law & medicine*, 39(4):497–538, 2013.
- [28] Wei Hu, Amrapali Zaveri, Honglei Qiu, and Michel Dumontier. Cleaning by clustering: methodology for addressing data quality issues in biomedical metadata. *BMC Bioinformatics*, 18(1):415, Sep 2017.
- [29] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Herve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.
- [30] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.
- [31] Purvesh Khatri, Silke Roedder, Naoyuki Kimura, Katrien De Vusser, Alexander A Morgan, Yongquan Gong, Michael P Fischbein, Robert C Robbins, Maarten Naesens, Atul J Butte, et al. A common rejection module (crm) for acute rejection across multiple organs identifies novel therapeutics for organ transplantation. *Journal of Experimental Medicine*, 210(11):2205–2221, 2013.
- [32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [33] Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762, 2014.
- [34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

- [36] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- [37] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [38] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [39] AZ Stuti Nayak, Amrapali Zaveri, and Michel Dumontier. Quality assessment of biomedical metadata using topic modeling. In *2nd workshop on Semantic Web solutions for large-scale biomedical data analytics (SeWeBMeDA)*, 2018.
- [40] Stuti Nayak, Amrapali Zaveri, Pedro Hernandez Serrano, Rachel Cavill, and Michel Dumontier. A machine learning approach towards quality assessment of biomedical metadata. *Journal of Biomedical Semantics*. Submitted November 2018.
- [41] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.
- [42] Maryam Panahiazar, Michel Dumontier, and Olivier Gevaert. Context aware recommendation engine for metadata submission. In *First International Workshop on Capturing Scientific Knowledge, Palisades, NY*, 2015.
- [43] Maryam Panahiazar, Michel Dumontier, and Olivier Gevaert. Predicting biomedical metadata in cedar: A study of gene expression omnibus (geo). *Journal of biomedical informatics*, 72:132–139, 2017.
- [44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [45] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [46] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

- [47] Lisa Posch, Maryam Panahiazar, Michel Dumontier, and Olivier Gevaert. Predicting structured metadata from unstructured metadata. *Database*, 2016, 2016.
- [48] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- [49] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333, 2011.
- [50] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
- [51] Fabrizio Sebastiani. Text categorization. In *Encyclopedia of Database Technologies and Applications*, pages 683–687. IGI Global, 2005.
- [52] V. Singh. Replace or Retrieve Keywords In Documents at Scale. *ArXiv e-prints*, October 2017.
- [53] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [54] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.
- [55] Brian Y Tsui, Shamim Mollah, Dylan Skola, Michelle Dow, Chun-Nan Hsu, and Hannah Carter. Creating a scalable deep learning based named entity recognition model for biomedical textual data by repurposing biosample free-text annotations. *bioRxiv*, page 414136, 2018.
- [56] W3C. Resource Description Framework (RDF). <http://www.w3.org/RDF/>, 2004.
- [57] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth and Carole Goble and Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester,

- Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3, 2016.
- [58] Zhu Xiaoliang, Yan Hongcan, Wang Jian, and Wu Shangzhuo. Research and application of the improved algorithm c4. 5 on decision tree. In *2009 International Conference on Test and Measurement*, volume 2, pages 184–187. IEEE, 2009.
 - [59] Antonio Jimeno Yepes and Rafael Berlanga. Knowledge based word-concept model estimation and refinement for biomedical text mining. *Journal of biomedical informatics*, 53:300–307, 2015.
 - [60] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational intelligence magazine*, 13(3):55–75, 2018.
 - [61] Amrapali Zaveri and Michel Dumontier. MetaCrowd: crowdsourcing gene expression metadata quality assessment. *F1000Research*, 6, 2017.
 - [62] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. Relation classification via convolutional deep neural network. 2014.
 - [63] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.
 - [64] Y-T Zhou, Rama Chellappa, Aseem Vaid, and B Keith Jenkins. Image restoration using a neural network. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1141–1151, 1988.
 - [65] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.