University of Kentucky
UKnowledge

Theses and Dissertations--Computer Science

Computer Science

2018

Deep Neural Networks for Multi-Label Text Classification: Application to Coding Electronic Medical Records

Anthony Rios
University of Kentucky, anthonymrios@gmail.com
Author ORCID Identifier:
https://orcid.org/0000-0003-1781-3975

Digital Object Identifier: https://doi.org/10.13023/etd.2018.306

Click here to let us know how access to this document benefits you.

Recommended Citation

Rios, Anthony, "Deep Neural Networks for Multi-Label Text Classification: Application to Coding Electronic Medical Records" (2018). *Theses and Dissertations--Computer Science*. 71.
https://uknowledge.uky.edu/cs_etds/71

This Doctoral Dissertation is brought to you for free and open access by the Computer Science at UKnowledge. It has been accepted for inclusion in Theses and Dissertations--Computer Science by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@lsv.uky.edu.

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Anthony Rios, Student

Dr. Ramakanth Kavuluru, Major Professor

Dr. Mirosław Truszczyński, Director of Graduate Studies

Deep Neural Networks for Multi-Label Text Classification: Application to Coding Electronic Medical Records

---

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Anthony Rios
Lexington, Kentucky

Director: Dr. Ramakanth Kavuluru, Associate Professor of Biomedical Informatics
Co-Director: Dr. Mirosław Truszczyński, Professor of Computer Science
Lexington, Kentucky 2018

Copyright© Anthony Rios 2018

ProQuest Number: 13843733

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 13843733

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

ABSTRACT OF DISSERTATION

Deep Neural Networks for Multi-Label Text Classification: Application to Coding Electronic Medical Records

Coding Electronic Medical Records (EMRs) with diagnosis and procedure codes is an essential task for billing, secondary data analyses, and monitoring health trends. Both speed and accuracy of coding are critical. While coding errors could lead to more patient-side financial burden and misinterpretation of a patients well-being, timely coding is also needed to avoid backlogs and additional costs for the healthcare facility. Therefore, it is necessary to develop automated diagnosis and procedure code recommendation methods that can be used by professional medical coders.

The main difficulty with developing automated EMR coding methods is the nature of the label space. The standardized vocabularies used for medical coding contain over 10 thousand codes. The label space is large, and the label distribution is extremely unbalanced – most codes occur very infrequently, with a few codes occurring several orders of magnitude more than others. A few codes never occur in training dataset at all.

In this work, we present three methods to handle the large unbalanced label space. First, we study how to augment EMR training data with biomedical data (research articles indexed on PubMed) to improve the performance of standard neural networks for text classification. PubMed indexes more than 23 million citations. Many of the indexed articles contain relevant information about diagnosis and procedure codes. Therefore, we present a novel method of incorporating this unstructured data in PubMed using transfer learning. Second, we combine ideas from metric learning with recent advances in neural networks to form a novel neural architecture that better handles infrequent codes. And third, we present new methods to predict codes that have never appeared in the training dataset. Overall, our contributions constitute advances in neural multi-label text classification with potential consequences for improving EMR coding.

KEYWORDS: Natural Language Processing, Machine Learning, Neural Networks, Multi-label Classification, Biomedical Informatics, Zero-shot Learning

Author's signature: Anthony Rios

Date: July 26, 2018

Deep Neural Networks for Multi-Label Text Classification: Application to Coding Electronic Medical Records

By
Anthony Rios

Director of Dissertation: Ramakanth Kavuluru

Co-Director of Dissertation: Mirosław Truszczyński

Director of Graduate Studies: Mirosław Truszczyński

Date: July 26, 2018

Dedicated to my parents, Armando and Gail, my wife, Gabriela, and my tiny puppy, Milo.

ACKNOWLEDGMENTS

First and foremost, I want to thank my advisor, Dr. Ramakanth Kavuluru, for his continuous support, guidance, and patience. Under his direction, I have learned to write, present, and perform research. Furthermore, his scientific knowledge and attention to detail has been an inspiration to me.

I would also like to thank my co-advisor and committee members: Drs. Mirosław Truszczyński, Judy Goldsmith, Nathan Jacobs, and Himanshu Thapliyal. Furthermore, I thank Dr. Richard Charnigo for his participation in my committee for my Ph.D. proposal. Their insights and comments have been invaluable in the preparation of my dissertation. Likewise, I would also like to thank Dr. Zhiyong Lu, my internship mentor at the National Library of Medicine, for providing me with the freedom, flexibility, and collaborative environment to pursue new research directions.

I also want to acknowledge my wife, Dr. Gabriela Romero Uribe, for her help reviewing my papers, posters, and presentations. The advice I have received from her has been vital to my success during my time in graduate school. She has provided me with a sympathetic ear during stressful times. Moreover, she has provided me her love, support, and encouragement which has been an essential part of my success.

Finally, I want to thank my parents, Armando and Gail Rios, for supporting me throughout my time as a student. Without their love and support, I would have not been able to attend graduate school. They have always kept me centered and let me know I can do anything I set my mind to.

Please note that this list is not comprehensive. Therefore, I want to thank all my friends and family who contributed to or have been a part of my life during my collegiate career. Thank you!

TABLE OF CONTENTS

Acknowledgments . . . iii

Table of Contents . . . iv

List of Tables . . . vii

List of Figures . . . viii

Chapter 1 Medical Coding . . . 1
- 1.1 Thesis statement . . . 4
- 1.2 Organization . . . 5
- 1.3 Related Publications . . . 6

Chapter 2 Related Work and Background . . . 7
- 2.1 Multi-label Classification . . . 7
- 2.2 Extreme Multi-label Classification . . . 8
- 2.3 Neural Networks for Text . . . 9
- 2.4 Diagnosis Coding and Biomedical Text Classification . . . 9
- 2.5 Notation . . . 10
- 2.6 Evaluation Measures . . . 11
- 2.7 EMR coding using Linear Models . . . 13
  - 2.7.1 Datasets . . . 14
  - 2.7.2 Multi-Label Text Classification Approaches . . . 16
  - 2.7.3 Results and Discussion . . . 21
  - 2.7.4 Conclusion . . . 24
- 2.8 Convolutional Neural Networks for Text Classification . . . 24
  - 2.8.1 Convolutions and CFs . . . 25
  - 2.8.2 Model Specification . . . 26
  - 2.8.3 Discussion . . . 29

Chapter 3 Extracting Diagnosis Codes using Transfer Learning from EMRs 30
- 3.1 Related Work: Transfer Learning . . . 32
- 3.2 Materials and Methods . . . 33
  - 3.2.1 Overview . . . 35
  - 3.2.2 Convolutional Neural Networks for Text Classification . . . 35
  - 3.2.3 Stage 1: Training on Source . . . 36
  - 3.2.4 Stage 2: Transfer Learning . . . 36
  - 3.2.5 Word Dropout . . . 38
  - 3.2.6 Ensemble . . . 38
- 3.3 Experiments . . . 38
  - 3.3.1 Implementation Details . . . 39

3.3.2 Baseline Methods . . . 39
3.3.3 Layer by Layer Analysis . . . 39
3.3.4 Comparison with prior work . . . 40
3.3.5 Label Frequency Analysis . . . 41
3.4 Conclusion . . . 41

Chapter 4 EMR Coding with Semi-Parametric Multi-Head Matching Networks 43
4.1 Related Work: Memory Augmented Neural Networks . . . 44
4.2 Our Architecture . . . 45
4.2.1 Convolutional Neural Networks . . . 46
4.2.2 Multi-Head Matching Network . . . 46
4.2.3 MetaLabeler . . . 48
4.2.4 Training . . . 48
4.2.5 Matching Network Interpretation . . . 49
4.3 Experiments . . . 50
4.3.1 Datasets . . . 50
4.3.2 Implementation Details . . . 51
4.3.3 Baseline Methods . . . 52
4.3.4 Evaluation Metrics . . . 52
4.3.5 Results . . . 52
4.4 Conclusion . . . 54

Chapter 5 Zero-shot and Few-shot Multi-label Learning . . . 55
5.1 Related Work . . . 58
5.1.1 Few-Shot and Zero-Shot Learning . . . 58
5.1.2 Structured Label Correlations for Multi-label Classification . . 59
5.1.3 Graph Convolutional Neural Networks . . . 59
5.2 Method . . . 60
5.2.1 Convolutional Neural Network . . . 61
5.2.2 Label Vectors . . . 61
5.2.3 Label-Wise Attention . . . 61
5.2.4 GCNN Output Layer . . . 62
5.2.5 Training . . . 63
5.3 Experiments . . . 63
5.3.1 Datasets . . . 64
5.3.2 Implementation Details. . . . 65
5.3.3 Evaluation Measures . . . 65
5.3.4 Baseline Methods . . . 65
5.3.5 Results . . . 66
5.4 Conclusion . . . 69

Chapter 6 Conclusion and Future Work . . . 70
6.1 Summary of Contributions . . . 70
6.2 Limitations and Directions for Future Work . . . 71

Appendix A Graph Regularized Concept Vectors 74

Abbreviations 77

Bibliography 80

Vita 92

LIST OF TABLES

2.1 CMC test set scores 22
2.2 Testing set scores for the UKYSmall dataset with BNS based feature selection. 23
2.3 Learning Component ablation for the UKYSmall dataset 23

3.1 Transfer learning dataset statistics. 34
3.2 Layer-by-layer results for various transfer learning methodologies. 40
3.3 Results comparing conventional approaches, CNNs, and CNNs with transfer learning 40

4.1 This table presents the number of training examples (# Train), the number of test examples (# Test), label cardinality (LC), and the average number of instances per label (AI/L) for the MIMIC II and MIMIC III datasets. 50
4.2 Results for the MIMIC II dataset. Models marked with * represent our custom implementations. 53
4.3 Results for the MIMIC III dataset. Models marked with * represent our custom implementations. 53
4.4 Ablation results for the MIMIC III dataset. 54

5.1 MIMIC II and MIMIC III dataset statistics for few- and zero-shot learning. 64
5.2 MIMIC II results across frequent (S), few-shot (F), and zero-shot (Z) groups. We mark prior methods for MIMIC datasets that we implemented with a *. 67
5.3 MIMIC III results across frequent (S), few-shot (F), and zero-shot (Z) groups. We mark prior methods for MIMIC datasets that we implemented with a *. 67
5.4 P@k, R@k, and macro-F1 results over all labels (the union of S, F, and Z). 68

LIST OF FIGURES

1.1 Example snippet from a discharge report in the MIMIC III dataset. . . . 3
1.2 Distribution of ICD-9 codes in the entire MIMIC III dataset. 4

2.1 Traditional CNN model for text classification which consists of three main components: A convolutional layer, max-over-time pooling, and a final output layer. 25

3.1 In Figure 3.1a, we show an example title and abstract from the Medline indexed paper by Foo et al. (2017). In Figure 3.1b, we show an example snippet from a discharge summary in the MIMIC III dataset (Johnson et al., 2016). 31
3.2 ICD-9 code frequency distribution of the UKYLarge EMR dataset. . . . 34
3.3 The parameters learned in the source task are transferred to the target task model and fixed while the target task specific model parameters are updated during training. 35
3.4 This figure compares the macro F-scores on the 10% least frequent codes to the macro F-score on the 10% most frequent ICD-9 diagnosis codes. . 41

4.1 The matching CNN architecture. For each input instance, **x**, we search a support set using different representations of **x** and use the similar support instances and auxiliary features to the output layer. 45

5.1 This plot shows the label frequency distribution of ICD-9 codes in MIMIC III. 56
5.2 Overview of our architecture 60
5.3 This graph plots the MIMIC III R@k for few-shot (F) labels at different k values. 68

6.1 Schematic for taking advantage of all the available structured and unstructured information available in PubMed and UMLS. 72

Chapter 1 Medical Coding

Electronic medical record (EMR) coding is the process of extracting diagnosis and procedure codes from the digital record (the EMR) about a patient's visit. The digital record is mostly composed of multiple textual narratives (e.g., discharge summaries, pathology reports, progress notes) authored by healthcare professionals, typically doctors, nurses, and lab technicians. Hospitals heavily invest in training and retaining professional EMR coders to annotate all patient visits by reviewing EMRs manually. Proprietary commercial software tools often termed as computer-assisted coding (CAC) systems are already in use in many healthcare facilities and were found to be helpful in increasing medical coder productivity (Dougherty et al., 2013). Thus, progress in automated EMR coding methods is expected to impact real-world operations for healthcare facilities.

In the United States, the diagnosis and procedure codes used for EMR coding are from the International Classification of Diseases (ICD) terminology (specifically the ICD-10-CM variant) as required by the Health Insurance Portability and Accountability Act (HIPPA). ICD codes facilitate billing activities, retrospective epidemiological studies, and also enable researchers to aggregate health statistics and monitor health trends. To code EMRs effectively, medical coders are expected to have thorough knowledge of ICD-10-CM and follow a complex set of guidelines. For example, if a coder accidentally uses the code "heart failure" (ICD-10-CM code I50) instead of "acute systolic (congestive) heart failure" (ICD-10-CM code I50.21), then the patient may be charged substantially more[1], causing significant unfair burden. Therefore, it is important for coders to have better tools at their disposal to find the most appropriate codes. In 2010, the Society of Actuaries sponsored a study that estimated the annual cost of medical coding errors is between 17 to 29 billion dollars (Shreve et al., 2010). If we can reduce the number of errors created by medical coders, then we can make a significant impact on the cost of healthcare. Additionally, if coders become more efficient, hospitals may hire fewer coders to reduce their operating costs. Thus, automated coding methods are expected to help with expedited coding, cost savings, and error control.

Discharge summaries are a common textual narrative used for medical coding. In the popular Medical Information Mart for Intensive Care (MIMIC) datasets (Lee

---

[1] `https://nyti.ms/2oxrjCv`

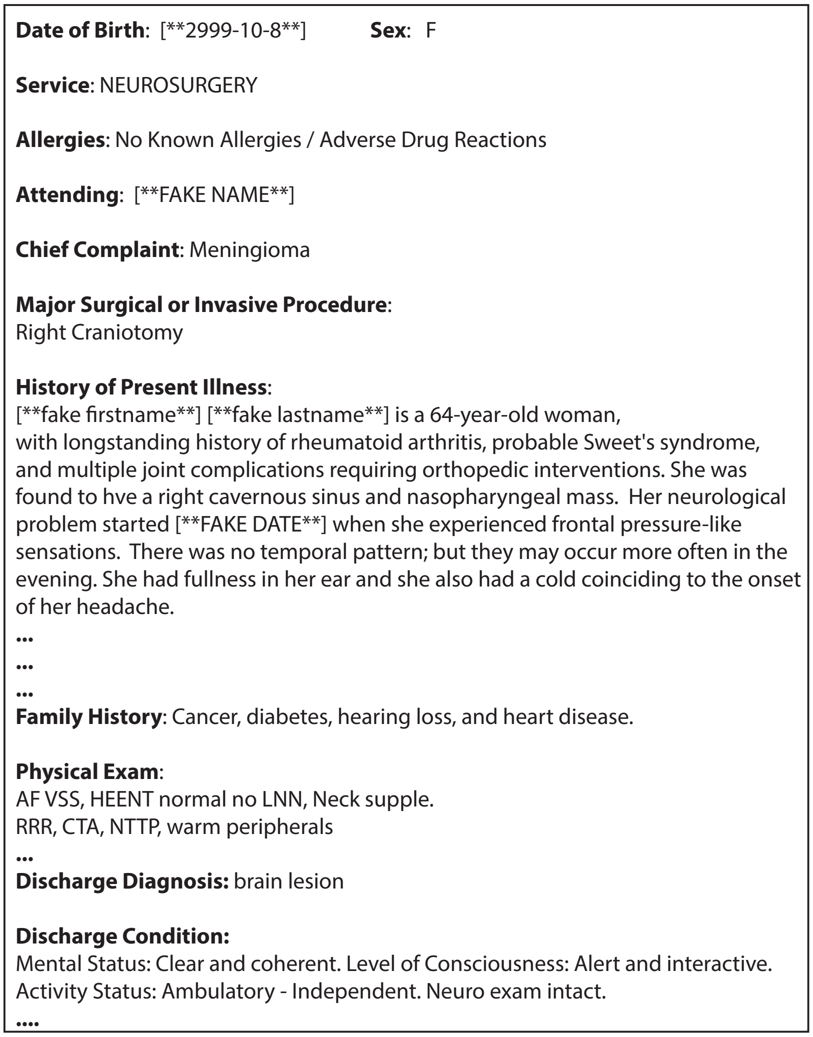
et al., 2011; Johnson et al., 2016), discharge summaries are the primary narratives available for developing automated medical coding systems. In the 2010 safe practices report released by the National Quality Forum (Meyer et al., 2010), they recommend the following information to be included in discharge summaries:

- the principal diagnosis and reason for visiting the hospital,
- significant findings for the patients visit,
- procedures performed and care, treatment, and services provided to the patient,
- the patients condition when they leave the hospital,
- information provided to the patient and family,
- a comprehensive and reconciled medication list, and
- a list of acute medical issues, tests, and studies for which confirmed results are unavailable at the time of discharge and require follow-up.

In Figure 1.1 we show a snippet of an example discharge summary from the MIMIC dataset. Overall, the report has a semi-structured format. Fields such as "Chief Complaint" and "Discharge Diagnosis" are easily parsed. However, fields such as "History of Present Illness" contain unstructured text data. The overall format of a MIMIC discharge report is relatively clean compared to our real-world EMRs which are generally more disordered. For example, in our dataset of medical records from the University of Kentucky (UKY), on average, EMRs contain more than 5000 words, which are around five times more words per instance compared to the MIMIC datasets. This is expected, given the UKY EMRs contain more information than a discharge report. Specifically, besides the discharge summary, UKY EMRs also contain progress notes, pathology, and radiology reports. We study both datasets in this dissertation.

We treat medical coding of EMR narratives as a multi-label text classification problem. Multi-label classification is a machine learning task that assigns a set of labels (typically from a fixed terminology) to an instance. Multi-label classification is different from multi-class problems, which assign a single label to each example from a set of three or more, and binary classification which assigns a single label from a set of two. Compared to general multi-label problems, EMR coding has three distinct challenges. First, with thousands of ICD codes, the label space is large and the label distribution is extremely unbalanced – most codes occur infrequently with a few codes occurring several orders of magnitude more than others. Figure 1.2 shows the

Figure 1.1: Example snippet from a discharge report in the MIMIC III dataset.



Date of Birth: [**2999-10-8**] **Sex**: F

Service: NEUROSURGERY

Allergies: No Known Allergies / Adverse Drug Reactions

Attending: [**FAKE NAME**]

Chief Complaint: Meningioma

Major Surgical or Invasive Procedure:
Right Craniotomy

History of Present Illness:
[**fake firstname**] [**fake lastname**] is a 64-year-old woman, with longstanding history of rheumatoid arthritis, probable Sweet's syndrome, and multiple joint complications requiring orthopedic interventions. She was found to hve a right cavernous sinus and nasopharyngeal mass. Her neurological problem started [**FAKE DATE**] when she experienced frontal pressure-like sensations. There was no temporal pattern; but they may occur more often in the evening. She had fullness in her ear and she also had a cold coinciding to the onset of her headache.

...

...

...

Family History: Cancer, diabetes, hearing loss, and heart disease.

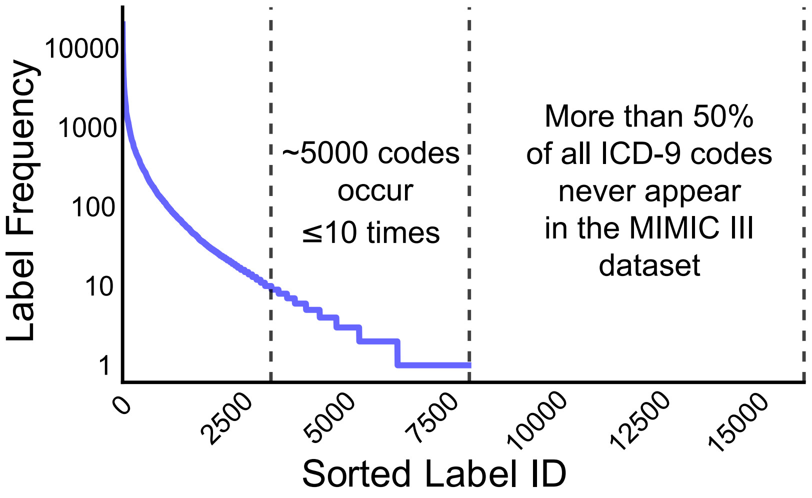
Physical Exam:
AF VSS, HEENT normal no LNN, Neck supple.
RRR, CTA, NTTP, warm peripherals
...

Discharge Diagnosis: brain lesion

Discharge Condition:
Mental Status: Clear and coherent. Level of Consciousness: Alert and interactive.
Activity Status: Ambulatory - Independent. Neuro exam intact.
....

distribution of diagnosis and procedure codes in the entire MIMIC III dataset. We find that around 5000 codes in the dataset occur less than ten times. Furthermore, MIMIC III uses the ICD-9-CM standardized terminology, which contains over 14 thousand unique diagnosis and procedure codes. From Figure 1.2 we find that more than half of all ICD-9-CM codes never occur in the dataset. While there have been great advances in machine learning-based medical coding methods (Baumel et al., 2017; Vani et al., 2017; Mullenbach et al., 2018), all methods completely ignore the fact that their systems will never predict an unseen ICD-9-CM code. Second, a patient may have a large number of diagnoses and procedures. On average, coders annotate an EMR with more than 20 such codes, hence predicting the top one or two codes is not sufficient for EMR coding. Moreover, combined with the problem of infrequently occurring codes, sophisticated multi-label thresholding techniques are required. Third, as previously

Figure 1.2: Distribution of ICD-9 codes in the entire MIMIC III dataset.



discussed, EMR narratives may be very long (i.e., discharge summaries alone may have over 1000 words), which may result in a needle in a haystack situation when attempting to seek evidence for particular codes. Therefore, developing methods that can be efficiently trained, and that can find the relevant information for all codes in a large document is important.

In this dissertation, we present neural network-based methods that solve the data sparsity issue and handle long documents. Neural networks have produced state-of-the-art performance on a wide variety of text classification problems (Collobert et al., 2011; Kim, 2014; Rios and Kavuluru, 2017). It is traditionally understood that neural networks require a lot of data to perform well. However, recently special architectures have been proposed that perform well on datasets with only a few examples per label (Vinyals et al., 2016; Snell et al., 2017b). It is also possible to take advantage of domains where data is abundant. For example, in Chapter 3 we take advantage of biomedical articles indexed by the PubMed search engine to augment a relatively small dataset of EMRs.

1.1 Thesis statement

Coding of EMRs is of immense importance for hospital and insurance providers. Machine learning-based methods can be used to create tools to improve the efficiency of medical coders and help reduce coding errors. In this dissertation, we are concerned with two significant challenges encountered when developing automated medical coding methods: data sparsity and handling long documents. We hypothesize that there are three uniques ways of handling these issues. First, if we augment our EMRs with biomedical textual datasets, then we can improve performance on infrequent ICD codes. Second, traditional neural networks generally require a large amount of data. We believe there are matching-based neural network architectures that can better extract ICD codes in the absence of large amounts of training data. Third, we believe that the inclusion of distributional and ontological information will lead to non-trivial performance gains on infrequent codes and codes that have never appeared in the training dataset.

1.2 Organization

The chapters in this dissertation are organized as follows:

Chapter 2 introduces relevant related work shared among all the methods in this dissertation. This chapter introduces the basic formal notation used to describe the various neural network-based methods throughout this manuscript. We then introduce the evaluation metrics used for multi-label classification. Next, we present our preliminary work using linear models to predict ICD codes to provide a basic understanding of classic methods. We conclude the chapter by providing a brief introduction to convolutional neural networks (CNN) for text classification.

Chapter 3 begins by providing an overview of work related to transfer learning, with an emphasis on methods used for natural language processing. Next, we introduce a simple, yet effective, method that overcomes the issue of "catastrophic forgetting", often associated with transfer learning. Furthermore, we provide detailed experiments on a real-world dataset of EMRs from the University of Kentucky, showing substantial improvements over prior work.

Chapter 4 describes a novel neural network architecture that incorporates ideas from memory networks (Vinyals et al., 2016; Rios and Kavuluru, 2018) to overcome the data sparsity problem. This chapter contains substantial experiments showing state-of-the-art performance on both the MIMIC II and MIMIC III datasets. We also provide a detailed ablation study to understand what aspects of the architecture have the largest impact on the overall performance.

Chapter 5 While Chapters 3 and 4 indirectly try to solve the data sparsity issue, neither of these methods are able to predict ICD-9-CM codes that do not occur in the training dataset. This chapter introduces an explicit method for few- and zero-shot classification. We also adapt a recently proposed evaluation methodology for generalized zero-shot learning to the multi-label domain. The method introduced in the chapter substantially outperforms prior methods for extracting ICD-9 codes on the MIMIC II and MIMIC III datasets with respect to few-shot codes. Furthermore, we show substantial improvements over traditional zero-shot methods.

Chapter 6 concludes the dissertation by summarizing the important contributions and results. Also, we describe a few promising avenues of exploration to improve our medical coding methods. One avenue focuses on extending our transfer learning work presented in Chapter 3. Another avenue ties our zero-shot work to open classification and proposes methods of making zero-shot classification useful to medical coders.

1.3 Related Publications

This dissertation contains material previously published in the following papers:

- **Anthony Rios** and Ramakanth Kavuluru. Supervised extraction of diagnosis codes from EMRs: role of feature selection, data selection, and probabilistic thresholding. In Proceedings of the IEEE International Conference on Healthcare Informatics (ICHI) 2013 (pp. 66–73).

- **Anthony Rios** and Ramakanth Kavuluru. Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Health Informatics (BCB) 2015 (pp. 258–267).

- **Anthony Rios** and Ramakanth Kavuluru. Analyzing the moving parts of a large-scale multi-label text classification pipeline: experiences in indexing biomedical articles. In proceedings of the IEEE International Conference on Healthcare Informatics (ICHI). 2015 (pp. 1–7).

- **Anthony Rios** and Ramakanth Kavuluru. EMR Coding with Semi-Parametric Multi-Head Matching Networks. In proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL) 2018 (pp. 2081–2091).

Chapter 2 Related Work and Background

In this Chapter, we discuss work that is related to all methods presented in this dissertation. We also present a formal introduction to the notation used in this manuscript, evaluation metrics for multi-label classification, discuss our prior work using linear models, and provide a brief introduction to neural networks for text classification. Related work that is only pertinent to a specific method is presented in its corresponding chapter.

2.1 Multi-label Classification

We begin by providing a brief review of traditional multi-label classification methods. Overall, there are two main class of multi-label methods. There is a class of methods called "problem transformation" approaches which convert the multi-label problem into multiple single-label classification problems. The second class of methods adapts a specific algorithm for single-label classification problems to directly predict multiple labels. This class of methods is generally called "algorithm adaptation". Both techniques, problem transformation and algorithm adaptation, are covered in a recent survey by Carvalho and Freitas (2009). Recent attempts in multi-label classification also consider label correlations when building a model for multi-label data (Read et al., 2011; Zhang and Zhang, 2010; Read et al., 2008). An important challenge in problems with a large number of labels per document is to decide the number of labels each document should be annotated with at test time, which has been addressed by calibrated ranking (Fürnkranz et al., 2008) and probabilistic thresholding (Quevedo et al., 2012). Feature selection is an important aspect when building traditional classifiers using machine learning. Forman (2003) provide a detailed comparative analysis of feature selection methods. In the multi-label scenario, the best set of features for one label may not be the best for another. However, performing feature selection for every label in large label spaces is infeasible. Therefore, methods which combine the feature relevance scores for each label have been proposed to find the best set of features jointly overall labels (Olsson, 2006). When dealing with datasets with class imbalance, methods such as random under/over-sampling, synthetic training sample generation, and cost-sensitive learning were proposed (see He and Garcia (2009) for a survey). In contrast with these approaches, Sohn et al. (2008) propose an alternative Bayesian method to curate customized optimal training sets for each label.

2.2 Extreme Multi-label Classification

“Extreme” in “extreme multi-label classification” is simply another term for “large-scale”. Large-scale can mean anything in the range from a few hundred to a million labels. Current methods for extreme multi-label classification fall into two categories: embedding and tree-based methods. Embedding-based methods aim to reduce the training complexity. They effectively reduce the label space by assuming the training label matrix is low rank. Intuitively, rather than learning independent classifiers for each label (binary relevance) (Tsoumakas et al., 2010), classifiers are learned in a reduced label space $\hat{L} \ll L$ where L is the total number of labels. Likewise, a projection matrix is learned to convert predictions from the reduced label space back to the original label space. In general, embedding methods vary based on how they reduce the label space and how the projection operation is optimized. Tai and Lin (2012) use principal component analysis (PCA) to reduce the label space. Low-rank Empirical risk minimization for Multi-Label Learning (LEML) (Yu et al., 2014) jointly optimizes the label space reduction and the projection processes. RobustXML (Xu et al., 2016) is similar to LEML but it treats infrequent labels as outliers and models them separately. Liu et al. (2017) employ neural networks for extreme multi-label problems using a funnel-like architecture that reduces the label vector dimensionality.

Tree-based multi-label methods work by recursively splitting the feature space. These methods usually differ based on the node splitting criterion. FastXML (Prabhu and Varma, 2014) partitions the feature space using the nDCG measure as the splitting criterion. PfastreXML (Jain et al., 2016) improves on FastXML by using a propensity scored nDCG splitting criterion and re-ranking the predicted labels to optimize various ranking measures.

While a lot of work has been proposed for large label spaces, simple traditional one-vs-rest linear models are still strong baselines, especially for textual data [1]. In our prior work, we show that sparse l1-regularized linear models can be combined into simple ensembles can provide state-of-the-art performance on large-scale biomedical text classification problems (Rios and Kavuluru, 2015a). Additionally, most extreme multi-label methods focus on ranking measures which give more weight to frequent labels rather than infrequent labels.

---

[1] `http://manikvarma.org/downloads/XC/XMLRepository.html`

2.3 Neural Networks for Text

Neural Networks have recently been adapted to natural language processing (NLP) tasks (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013b). Convolution neural networks (CNN) take advantage of the so called "convolutional filters" to automatically learn features that are suitable to the task at hand. They have been actively used in biomedicine in image classification, even before deep learning became popular recently. However, CNNs have not been explored until recently for text classification, and are currently used for sentiment/opinion mining (Kim, 2014; Kalchbrenner et al., 2014) for short texts with fairly balanced class distributions. CNNs have also made advances in other natural language processing (NLP) tasks including named entity recognition (Collobert et al., 2011) and relation extraction (Zeng et al., 2014; Santos et al., 2015).

2.4 Diagnosis Coding and Biomedical Text Classification

Several attempts have been made to extract ICD codes from clinical documents since the 1990s. Advances in natural language and semantic processing techniques contributed to a recent surge in automatic extraction. Lima et al. (1998) use a hierarchical approach utilizing the alphabetical index provided with the ICD-9-CM resource. Although completely unsupervised, this approach is limited by the index not being able to capture all synonymous occurrences, and also the inability to code both specific exclusions, and other condition-specific guidelines. We have also shown label hierarchical information can provide improvements on general biomedical text classification (Kavuluru and Rios, 2015). Gundersen et al. (1996) extracted ICD-9 codes from short free text diagnosis statements that were generated at the time of patient admission using a Bayesian network to encode semantic information. However, recently, concept extraction from longer documents such as discharge summaries has gained interest. Some advances are based on systems trained on the CMC dataset developed for the BioNLP workshop shared task on multi-label classification of clinical texts in 2007 (Pestian et al., 2007). We discuss our preliminary work using the CMC dataset in Section 2.7. Other methods have focused on more realistic datasets, in terms of size and the number of labels (Johnson et al., 2016). Comprised of approximately sixty thousand admissions of patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012, MIMIC is a real dataset and provides a strong test bed for machine learning methods that large datasets.

Traditionally, linear methods have been used for diagnosis code prediction and general biomedical text classification (Rios et al., 2013). Perotte et al. (2013) developed a hierarchical support vector machine (SVM) model that takes advantage of the ICD-9-CM hierarchy. In our prior work, we train a linear model for every label (Rios and Kavuluru, 2013) and re-rank the labels using a learning-to-rank procedure (Kavuluru et al., 2015). Zhang et al. (2017) supplement the diagnosis code training data with data from PubMed (biomedical article corpus and search system) to train linear models using both the original training data and the PubMed data. We apply a similar technique to neural networks in Chapter 3.

Recent advances in neural networks have also been put to use for EMR coding: Baumel et al. (2017) trained a CNN with multiple sigmoid outputs using binary cross-entropy. Duarte et al. (2017) use hierarchical recurrent neural networks (RNN) to annotate death reports with ICD-10 codes. Vani et al. (2017) introduced grounded RNNs for EMR coding. They found that iteratively updating their predictions at each time step significantly improved the performance. Finally, similar to our work presented in Chapter 4, memory networks (Prakash et al., 2017) have recently been used for diagnosis coding.

2.5 Notation

In this work, matrices are represented using bold upper case letters ($\mathbf{W}$, $\mathbf{U}$, $\mathbf{V}$) and vectors are represented as bold lower case letters ($\mathbf{w}$, $\mathbf{x}$, $\mathbf{y}$). Subscripts are used to distinguish different matrices and vectors ($\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{y}_1$, $\mathbf{y}_2$). If there are many matrices or vectors representing something very similar, then superscripts are used to distinguish them ($\mathbf{W}_i^1$, $\mathbf{W}_i^2$, $\mathbf{W}_i^3$). Finally, we use lower case letters to represent scalars (y).

Throughout this document, let $\mathbf{y}_i \in \{0,1\}^L$ be the ground-truth label vector for the i-th instance, where L is the size of the fixed label set. $\mathbf{y}_{i,j} = 1$ if and only if the i-th instance is assigned the j-th label; all other elements are zeros. $\mathbf{Y} \in \{0,1\}^{N \times L}$ is a label matrix of the true labels for N instances. Per instance label predictions are represented as $\hat{\mathbf{y}}_i$ and the prediction matrix is $\hat{\mathbf{Y}} \in \{0,1\}^{N \times L}$.

Finally, in general, we define specific notation and terms as required in each chapter. By defining notation as needed, individual chapters can be read in any order. We make an exception for material that each chapter has in common. If a specific component is common among all methods presented in this dissertation, then we define it once at its first appearance in the manuscript and reference it in future chapters.

Also, many variables may be reused across chapters; however dimensions may be different. At the first occurrence for each variable or in the chapters implementation details chapter, we explicitly state dimensions to avoid confusion. For example, the number of labels in the output layer dimensions may differ between chapters (i.e., Chapter 3 uses a different dataset compared to Chapters 4 and 5).

2.6 Evaluation Measures

For the purpose of this dissertation, two groups of evaluation measures are defined: the first works with *multi-label classification* where the goal is to predict a set of labels. The second pertains to *multi-label ranking*, which is suitable for problems with large label spaces, especially in the context label recommendation. Given an input instance, multi-label ranking promotes relevant labels to the top of the ranked label list.

For multi-label classification, micro and macro F-score are widely used (Tsoumakas et al., 2010). For each label l_j in the set of labels $\mathbf{l}$ being considered, we have label-based precision $P(l_j)$, recall $R(l_j)$, and F-score $F(l_j)$ defined as

$$P(l_j) = \frac{TP_j}{TP_j + FP_j}, \qquad R(l_j) = \frac{TP_j}{TP_j + FN_j},$$

$$\text{and } F(l_j) = \frac{2P(l_j)R(l_j)}{P(l_j) + R(l_j)}, \tag{2.1}$$

where TP_j, FP_j, and FN_j are true positives, false positives, and false negatives, for label l_j, respectively. Precision is the fraction of predicted relevant labels among all predicted labels, while recall is the fraction of predicted relevant labels among all relevant labels. Then F() is simply the harmonic mean between precision and recall for the j-th label. Given the label-wise F-scores defined in Equation 2.1, the label-based macro averaged F-score is defined as

$$\text{Macro-F} = \frac{1}{L}\sum_{j=1}^{L} F(l_j). \tag{2.2}$$

Intuitively, macro measures consider all labels equally independent of how many times each label occurs in the dataset.

The label-based micro precision, recall, and F-score are defined as

$$P^{mic} = \frac{\sum_{j=1}^{L} TP_j}{\sum_{j=1}^{L}(TP_j + FP_j)}, \qquad R^{mic} = \frac{\sum_{j=1}^{L} TP_j}{\sum_{j=1}^{L}(TP_j + FN_j)},$$

$$\text{and Micro-F} = \frac{2P^{mic} \cdot R^{mic}}{P^{mic} + R^{mic}}, \tag{2.3}$$

respectively. Unlike macro-based measures, micro measures tend to give more importance to labels that are more frequent.

The issue with both macro- and micro-based measures is that a predefined threshold must be set to predict each label. For example, assuming a logistic regression classifier, each label will have a score between 0 and 1. The default threshold used is typically 0.5. If we predict a specific label, then our classifier is more confident than not. It is also possible to use a different threshold. For example, if we increase our threshold, then we expect our recall to decrease while possibly increasing precision. Furthermore, different labels may require different thresholds. To avoid these issues we also use the area under the receiver operating characteristic curve (AUCOC). AUROC is calculated by first calculating the true positive rate (also known as recall) and the false positive rate

$$\frac{FP}{FP + TN}$$

using multiple thresholds ranging from 0 to 1. This produces a curve, then AUROC is calculated by measuring the area under the curve. Similar to micro-based measures, AUROC gives more weight to frequent classes. So, for imbalanced datasets, AUROC tends to be overly optimistic. Therefore, another metric is the area under the precision recall curve (AUPRC). Similar to AUROC, the recall is calculated across multiple thresholds. However, instead of using the false positive rate, AUPR uses precision. Both, AUROC and AUPRC can be calculated for each label individually. Likewise, both macro- and micro-based measures are possible with both metrics. For macro AUROC and AUPRC, the area under the curve is first calculated for each label, then the scores are simply averaged together similar to Equation 2.2. For micro-based scores the micro-based precision, recall, and false positive rates are calculated over all labels using the same threshold at each point in the curve.

The next set of metrics evalaute multi-label ranking. Precision at k (P@k) is a common evaluation measure for evaluating extreme multi-label ranking methods (Yu et al., 2014). Intuitively, $P@k$ weights the top k predictions equally. This means the absolute rank is not important as long as the labels are ranked in the top k.

Moreover, unlike precision for classification-based metrics, $P@k$ ignores the aboslute score. $P@k$ is defined as

$$P@k = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{k}\sum_{l\in rank_k(\hat{\mathbf{y}})}\mathbf{y}_{i,l}, \tag{2.4}$$

where $rank_k(\hat{\mathbf{y}})$ is the set of top k ranked label indices. Intuitively, the precision is calcualted for each example independently, then all the scores are averaged examples. This is known as a example-based measure. Similarly, we define the R@k as

$$R@k = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\|\mathbf{y}_i\|_0}\sum_{l\in rank_k(\hat{\mathbf{y}})}\mathbf{y}_{i,l} \tag{2.5}$$

where $\|\mathbf{y}_i\|_0$ represents the L0 norm on the ground truth label vector which counts the number of labels assigned to the i-th example. $R@k$ is a better measure when each example is only annotated with a few labels. For example, if we measure $P@10$ and each example has on average only 2 labels, the scores will be small even if the top 10 ranked labels contain all true positives.

Finally, we briefly define the traditional example-based precision, recall, and F-score measures as

$$P^{ex} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L} TP_j^i}{\sum_{j=1}^{L}(TP_j^i + FP_j^i)}, \qquad R^{ex} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L} TP_j^i}{\sum_{j=1}^{L}(TP_j^i + FN_j^i)},$$

$$\text{and Ex-Micro-F} = \frac{1}{N}\sum_{i=1}^{N}\frac{2P^{ex}\cdot R^{ex}}{P^{ex}+R^{ex}}, \tag{2.6}$$

where TP_j^i, FP_j^i, and FN_j^i represent the number of true positives, false positives, and false negatives for the i-th instance and the j-th label, respectively. Both P^{ex} and R^{ex} are similar to R@k and P@k. If the true positives, false positives, and false negatives are calculated using a ranking based thresholding strategy, then the two evaluation methodologies are equivalent. However, the traditional example-based measures generalize to other thresholding strategies.

2.7 EMR coding using Linear Models

Before we provide an introduction to neural networks (NN) for text classification, and before we present our NN methods for assigning ICD codes to EMRs, we describe our preliminary work using traditional linear models, feature selection, and data subset selection methodologies to extract ICD codes from small datasets. Moreover, we discuss the limitations of the methods and datasets described in the section.

2.7.1 Datasets

The 2007 shared task on coding radiology reports (Pestian et al., 2007) was the first effort that popularized automated EMR coding. The CMC dataset consists of 1954 radiology reports arising from the outpatient chest x-ray and renal procedures and is observed to cover a substantial portion of pediatric radiology activity. The radiology reports are formatted in XML with explicit tags for *history* and *impression* fields. There are a total of 45 unique codes and 94 distinct combinations of these codes in the dataset. The dataset is split into training and test sets of nearly equal size where reports for all possible codes and combinations occur in both sets. This means that all possible combinations that are encountered in the test set are known ahead of time. Furthermore, from the 1954 reports in the CMC dataset, 978 are included in the training dataset with their corresponding ICD-9 codes; the remaining documents form the test set. The top system on the CMC dataset obtained a micro-average F-score of 90% and 21 of the 44 participating systems scored between 80% and 90%.

For each instance in the CMC dataset there are two fields (in the XML file) that contain textual data, 'clinical history' and 'impression'. The clinical history field contains textual information entered by a physician before a radiological procedure about the patient's history. The impression field contains textual information entered by a radiologist after the radiological procedure about his observations of the patient's condition. Many of the entries contained in these two fields are very short. An example clinical history field is "22 month old with cough." The corresponding impression is just one word "normal." The average size of a report is only 21 words.

For this study, we also built a second dataset to study code extraction at the EMR level, UKYSmall . We created a dataset of 1000 clinical documents corresponding to a randomly chosen set of 1000 in-patient visits to the UKY Medical Center in the month of February, 2012[2] We also collected the ICD-9 codes for these EMRs assigned by a trained coder at the UKY medical records office. Aggregating all billing data, this dataset has a total of 7480 diagnoses leading to 1811 unique ICD-9 codes. ICD-9 codes have the format `abc.xy` where the first three digits `abc` capture the category. Using the (category code, label, count) representation, the top 5 most frequent categories

---

[2] This dataset has been approved by the UKY IRB for use in research projects (protocol #12-0139-p3h).

are (`401`, *essential hypertension*, 325), (`276`, *Disorders of fluid electrolyte and acid-base balance*, 239), (`305`, *nondependent abuse of drugs*, 236), (`272`, *disorders of lipoid metabolism*, 188), and (`530`, *diseases of esophagus*, 169). On average, each EMR is annotated with 7.5 codes. It is important to understand that each EMR can be assigned a different number of ICD-9 codes. There are EMRs with only one code, while the maximum number of ICD-9 codes assigned to an EMR is 49. For each in-patient visit, the original EMR consisted of several documents, some of which are not plain text files but are stored in the RTF format. Some documents, like care flowsheets, vital signs sheets, and ventilator records were not considered for this analysis. Overall, we have a total of 5583 documents for all 1000 EMRs. There is an average of 5.6 textual documents per EMR, but we only consider those authored by physicians. After filtering out documents not authored by physicians, we obtain an average of 2.8 documents per EMR.

Since many of the 1811 codes in the UKYSmall dataset are assigned to very few examples, for this preliminary study, we decided to consider extracting codes at the fourth digit level. In other words, all codes of the form `abc.xy` for different `‘y’` are mapped to the four digit code `abc.x`. With this mapping, the number of unique codes is reduced to 1410. Note the the average number of codes per EMR even when we collapsed the fifth digit is still 7.5 (the same value as for the case of full five digit codes). This is because, in general, an EMR does not have two codes that differ at the fifth digit, which captures the finest level of classification. The average size of each EMR (that is, of all textual documents in it) in the UKYSmall dataset is 2088 words. Even when truncated to 4 digits, there were still many codes that had too few examples to apply traditional supervised methods. Hence, we resorted to using only 56 codes (at that 4th digit level) that had at least 20 EMRs in the dataset; the number of unique combinations of these 56 codes is about 554. After removing those EMRs that did not have any of these frequently occurring 56 codes, we are left with 827 EMRs in the dataset. We randomly select and removed 100 examples from the dataset to be used for testing and the remaining 727 EMRs form the training dataset.

Label-cardinality is the average number of codes per report (or EMR in the UKYSmall dataset). To use consistent terminology we refer to the single reports in the CMC dataset as EMRs that consist of just one document. Let m be the total number of EMRs and $\mathcal{Y}_i$ be the set of labels for the i-th EMR. Label-cardinality is defined as

$$\text{Label-Cardinality} = \frac{1}{m}\sum_{i=1}^{m} |\mathcal{Y}_i|.$$

The CMC training dataset has a label cardinality of 1.24 and the UKYSmall dataset, counting on the 56 frequent codes, has a label cardinality of 3.86.

The CMC and UKYSmall datasets have significant differences: the CMC data set is coded by three different coding companies and final codes are consolidated from these three different extractions. As such, it is of higher quality compared to the UKYSmall dataset which is coded by only one trained coder from the UKY medical records office. On the other hand, the CMC dataset does not have the broad coverage of the UKYSmall dataset, which models a more realistic dataset at the EMR level. The CMC dataset only includes radiology reports and has 45 codes with 94 code combinations and has on an average 21 words per EMR. In contrast, even with the final set of 56 codes (at the four digit level) that have at least 20 examples that we use for our experiments, the number of combinations for the UKYSmall dataset is 554 with the average EMR size two orders of magnitude more than the average for the CMC dataset. However, the number of codes and the size of the dataset in the UKYSmall dataset is not realistic compared to the experiments with larger datasets in Chapters 3, 4, and 5.

2.7.2 Multi-Label Text Classification Approaches

In this section, we describe the methods we employ for multi-label classification to extract ICD-9 codes from both of our datasets. Our core methods primarily use the problem transformation approach where the multi-label classification problem is converted into multiple binary classification problems, binary relevance. We also use different approaches that take into account label correlations expressed in the training data. Besides this basic framework, we utilize feature selection, training data selection, and probabilistic thresholding as additional components of our classification systems. One of the goals of this preliminary work is to see how all these components and transformation approaches perform on the two different medical coding datasets, CMC and UKYSmall.

Document features. We use unigram and bigram counts as features. Stop words (determiners, prepositions, and so on) were removed from the unigrams. Since unigrams and bigrams are syntactic features, we also use semantic features such as *named entities* and binary relationships (between named entities) extracted from text, typically called *semantic predications.*

To extract named entities and semantic predications, we use software made available through the Semantic Knowledge Representation (SKR) project by the National

Library of Medicine (NLM). Specifically, the two software packages we use were MetaMap and SemRep. MetaMap (Aronson, 2001) is a biomedical named entity recognition program that identifies concepts from the Unified Medical Language System (UMLS) Metathesaurus, an aggregation of over 160 biomedical terminologies. When MetaMap outputs different named entities, it associates a confidence score in the range from 0 to 1000. We only use concepts with a confidence score of at least 700 as features. Each of the concepts extracted by MetaMap also contain a field specifying if the concept was negated (e.g., "no evidence of hypertension"). We use negated concepts that capture the absence of conditions/symptoms as different features from the original concepts. We use SemRep (Rindflesch et al., 2003) , a relationship extraction program that extracts semantic predications of the form $C1 \rightarrow \ relationType \ \rightarrow C2$ where $C1$ and $C2$ are two different biomedical named entities and $relationType$ expresses a relation between them (e.g., "Tamoxifen *treats* Breast Cancer"). Because predication extraction is made by making calls using the Web API provided by SKR, we only use predications as features for the de-identified CMC dataset and not for the UKYSmall dataset. If there is more than one document in an EMR, features are aggregated from all documents authored by a physician.

Problem Transformations for Multi-Label Classification. For the binary classifiers for each label, we experimented with three base classifiers: Support Vector Machines (SVMs), Logistic Regression (LR), and Multinomial Naive Bayes (MNB). We used the MNB classifier that is made available as part of the Weka framework. For LR, we used LibLINEAR (Fan et al., 2008) implementation in Weka and for SVMs we used LibSVM (Chang and Lin, 2011) in Weka.

We experimented with four different multi-label multi-label problem transformation methods: *binary relevance, copy transformation, ensemble of classifier chains,* and *ensemble of pruned label sets.*

Let T be the set of all labels and let $L = |T|$. Binary relevance learns L binary classifiers, one for each label in T. It transforms the dataset into L separate datasets. For each label T_j, we obtain the dataset for the corresponding binary classifier by considering each document–label-set pair $(D_i, \mathcal{Y}_i)$ and generate the document-label pair (D_i, T_j) when $T_j \in \mathcal{Y}_i$ or the pair $(D_i, \neg T_j)$ when $T_j \notin \mathcal{Y}_i$. When predicting, the labels are ranked based on their score output by the corresponding binary classifiers and the top r labels are considered as the predicted set for a suitable r. The copy transformation transforms multi-label data into single-label data. Let $T = \{T_1, \dots, T_L\}$ be the set of L possible labels for a given multi-label problem. Let each document

$D_j \in D$, $j = 1, \ldots, m$, have a set of labels $\mathcal{Y}_j \subseteq T$ associated with it. The copy transformation transforms each document–label-set pair $(D_j, \mathcal{Y}_j)$ into $|\mathcal{Y}_j|$ document–label pairs (D_j, T_s), for all $T_s \in \mathcal{Y}_j$. After the transformation, each input pair for the classification algorithm will only have one label associated with it and one can use any single-label method for classification. The labels are then ranked based on the score given from the classifier when generating predictions. We then take the top r labels as our predicted label set.

One of the main disadvantages of the binary relevance and copy transformation methods is that they assume label independence. In practical situations, there can be dependence between labels when labels co-occur frequently or when a label occurs only when a different label is also tagged. Classifier chains (Read et al., 2011), based on the binary relevance transformation, try to account for these dependencies that the basic transformations cannot. Like binary relevance, classifier chains transform the dataset into L datasets for binary classification per each label. But they differ from binary relevance in the training phase. Classifier chains loop through each dataset in some predefined order, training each classifier one at a time. Each binary classifier in this order will add a new Boolean feature to the subsequent binary classifier datasets to be trained next. The issue with classifier chains, is that its performance depends heavily on the predefined order of the labels. Read et al. (2011) propose a simple, yet effective, ensemble method which overcomes the issue of label ordering, ensemble of classifier chains (ECC).

While the ensemble of classifier chains method overcomes some of the issues involving the ordering of chaining, for large label spaces, larger ensembles are required to overcome the label ordering issue. Another multi-label classification method that takes label correlations into account is the pruned label sets approach (Read et al., 2008). In this approach, a multi-label problem is transformed into a multi-class problem by representing combinations of different labels as new classes. Let each document $D_j \in D$, $j = 1, \ldots, m$, have a set of labels $\mathcal{Y}_j \subseteq T$ associated with it. Pruned sets will treat each unique set of labels among $\mathcal{Y}_j$ as a single class and corresponding EMRs as training examples for that class. Only combinations whose training data counts are above a user chosen threshold are converted into new classes. For infrequent combinations, smaller subsets of these combinations that are more frequent are converted into new classes with training examples obtained from the corresponding combinations. To overcome the issue of not being able to predict subsets of very frequent combinations (since they are already converted into separate classes) in the basic pruned sets approach, using an ensemble approach, several pruned set classifiers (Read et al., 2008) are trained on random subsets of the original training data with final predictions made using a voting approach for each label.

Feature selection. An important issue in classification problems with a large number of number of classes and features is that the features most relevant in classifying one class from the rest might not be the same for every class. Furthermore, many features are either redundant or irrelevant to the classification tasks at hand. In the domain of text classification, Bi-Normal Separation (BNS) score was observed to result in best performance in most situations among a set of 12 feature selection methods applied to 229 text classification problems (Forman, 2003). We employ this feature scoring method for this effort. Let $T = \{T_1, \ldots, T_L\}$ be the set of L possible labels. For each label $T_i \in T$ and for each feature, we calculate the number of true positives (tp) and false positives (fp) with respect to that feature – tp is the number of training EMRs (with label T_i) in which the feature occurs. Similarly, fp is the number of negative examples for T_i in which the feature occurs. Let pos and neg be the total number of positive and negative examples for T_i, respectively. With F^{-1} denoting the inverse cumulative probability function for the standard normal distribution, we define the BNS score of a given feature for a particular class as

$$BNS = |F^{-1}(tpr) - F^{-1}(fpr)| \qquad \text{where}$$

$$tpr = \frac{tp}{pos} \quad \text{and} \quad fpr = \frac{fp}{neg}.$$

Because $F^{-1}(0)$ is undefined, we set tpr and fpr to 0.0005 when tp or fp are equal to zero.

For each of the L binary classification problems, we pick the top k ranked features for each label. In our experiments $k = 8000$ gave the best performance out of a total of 68364 features for the UKYSmall dataset. The number of features was very small for the CMC dataset, with a total of 2296 features, feature selection did not improve the performance.

Greedy 'Optimal' Training Data Selection In the case of multi-label problems with a large number of classes, the number of negative examples is overwhelmingly larger than the number of positive examples for most labels. We experimented with the synthetic minority oversampling approach (Chawla et al., 2002) for the positive examples which did not prove beneficial for our task. Deviating from the conventional random under/over-sampling approaches, we adapted the 'optimal' training

Algorithm 1 Optimal Training Set Selection

```
for all Binary datasets $B_1$ to $B_q$ do
    Move 20% of the positive examples and 20% of the negative examples from $B_i$ to a validation dataset ($V_i$).
    Put the remaining positive examples into a smaller training dataset ($STS_i$).
    Score the remaining negative examples in $B_i$ according to their similarity with positive examples.
    Initialize snapshot variable $k = 1$
    while $B_i$ is not empty do
        Remove the top 10% scored negative examples in $B_i$ and add them to $STS_i$.
        Record the snapshot of the current training set, $STS_i^k = STS_i$.
        Build a binary classifier for $i$-th code with training dataset $STS_i^k$ and record the $F_{1.5}$ score on $V_i$.
        $k = k + 1$
    Set the optimal training set $OTS$ = the snapshot $STS_i^k$ with the highest $F_{1.5}$ score.
```

set (OTS) selection approach used for medical subject heading (MeSH terms) extraction from biomedical abstracts by Sohn et al. (2008). The OTS approach is a greedy Bayesian method that under-samples the negative examples to select a customized dataset for each label. The greedy selection is not technically optimal but we stick with the terminology in (Sohn et al., 2008) for clarity. The method for finding OTS is described in Algorithm 1. Intuitively, the method ranks negative examples according to their similarity to positive examples and iteratively selects negative examples according to this ranking, and finally selects the negative subset that offers the best performance on a validation set for that label.

Here we describe how to compute the score for negative examples. Since we want to select those negative examples that are the most difficult to distinguish from the positive examples, for a given negative training example D_j, we would like our score to be proportional to $P(positive|D_j)$. Using Bayes theorem, this quantity can be shown (see the online appendix A of (Sohn et al., 2008)) to monotonically increase with the following score function

$$Score(D_j) = ln\left(\frac{P(D_j|\text{positive})}{P(D_j|\text{negative})}\right),$$

where $P(D_j|positive)$ is the probability estimate of document D_j given the class is positive, which is estimated using $\prod_i P(w_i|positive)$ where w_is are the unigrams in D_j. Each $P(w_i|positive)$ is estimated using counts of w_i in the positive examples in the training data. Similarly, $P(D_j|negative)$ is also estimated. The measure we use for assessing the performance of each STS on the validation set in algorithm 1 is the traditional F_β measure with $\beta = 1.5$. We chose $F_{1.5}$ over the traditional balanced F_1 measure to give more importance to recall since the main utility of code prediction is to assist trained coders. Our adaptation of the Sohn et al. (2008) method differs from the original approach in that we don't resort to the leave-one-out cross validation and instead use a validation set to select the OTS. We also use the naive Bayes assumption for estimates of $P(D_j|positive)$ involving terms for unigrams that occur in the document.

Probabilistic thresholding. In multi-label classification, it is also important to consider the effect of the number of labels predicted per document on the performance of the approaches used. The example, macro and, micro F-scores are dependent on the threshold because it can significantly impact the weight between precision and recall. A straightforward (and the default approach in many implementations) is to predict all labels whose base binary classifiers return posterior probabilities > 0.5. This could lead to more labels than is actually the case or fewer labels than the actual number. A quick fix that many employ is to pick a threshold of top r (r-cut) ranked labels where r is chosen to maximize the example-based F-score on the training data. However, simply picking the top r labels will always result in the same number of predicted labels for each document in the test set. We used an advanced thresholding method, Multi Label Probabilistic Threshold Optimizer (Quevedo et al., 2012) (MLPTO), for choosing a different number of labels per EMR that can vary for each EMR instance. The optimizer we employed uses $1 -$ (Example-Based-F-score) as the loss function and finds the r that minimizes the expected loss function across all possible example-based contingency tables for each instance. For specific details of this strategy, please see Quevedo et al. (2012).

2.7.3 Results and Discussion

In this section we present the results we obtain on both datasets and assess the role of different components of the learning approaches we explored in Section 2.7.2.

CMC dataset results. The best method on the CMC dataset are obtained using unigrams and named entity counts (without any weighting) as features and SVMs as the base classifiers. In Table 2.1, we present the results when using the binary relevance (BR), ensemble of classifier chains (ECC), and ensemble of pruned sets (EPS)

Table 2.1: CMC test set scores

		Example-Based			Micro			Macro		
		Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
BR	SVM	82	82	81	86	81	83	54	48	49
	SVM/Bagging	83	82	81	87	81	84	56	47	49
EPS	SVM	86	81	82	88	78	83	44	32	34
ECC	SVM	84	84	83	88	82	85	54	44	47

problem transformations. We also used SVMs with bagging to compare a simple BR-based ensemble with the more complex ensemble approaches ECC and EPS, which take label dependencies into account. We find that the best performing classifier is the ensemble of classifier chains with an F-score of 85%. It is interesting to see that BR with bagging also performed reasonably well. For the CMC competition, the micro F-score was the measure used for comparing relative performances of contestants. The mean score in the competition was 77% with a standard deviation of 13%. The best performing method was able to achieve a 90% micro F-score.

There were many instances in the CMC dataset where our methods did not predict any codes. We experimented with an unsupervised approach to generate predictions for those examples: we generated named entities using MetaMap for each of these documents that did not have any predictions. We mapped these entities to ICD-9-CM codes via a knowledge-based mapping approach (Bodenreider et al., 1998) that exploits the graph of relationships in the UMLS Metathesaurus (which also includes ICD-9-CM). If MetaMap generated a concept that got mapped to an ICD-9 code we trained on, we used that ICD-9 code as our prediction. We were able to increase our best F-score from using ECC from 85% to 86% using this method. Also, while we don't report the results for when we added semantic predications as features, the results were comparable with no major improvements. Feature selection, optimal training sets, and probabilistic thresholding did not make significant improvements for the CMC dataset.

UKYSmall dataset results. In Table 2.2 we present the results on the test set for the UKYSmall dataset. For this dataset, we first tried our best performing models from the CMC dataset. We noticed that ECC did not perform well on this larger dataset; there seem to have been very few label dependencies in this dataset – recall that there were over 500 unique label sets of the 56 labels used for training for the in house dataset. It is also possible that the method was not sophisticated enough to capture the label correlations. Also, on this dataset, we achieved the best results

Table 2.2: Testing set scores for the UKYSmall dataset with BNS based feature selection.

		Example-Based			Micro			Macro		
		Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
BR	LR	38	21	24	64	16	25	21	13	15
	LR+MLPTO+OTS	59	42	45	54	37	44	40	29	32
	LR+OTS+RCut 3	40	49	38	40	41	41	34	31	29
Copy	LR+RCut 3	39	49	38	39	39	39	30	32	28

Table 2.3: Learning Component ablation for the UKYSmall dataset

	Example-Based			Micro			Macro		
BR/LR (common)	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
BNS+OTS+MLPTO	59	42	45	54	37	44	40	29	32
-MLPTO	52	34	37	61	32	42	43	26	30
-OTS	60	36	40	57	29	39	42	25	29
-BNS	30	11	15	31	10	15	05	04	04

using LR instead of an SVM as our base classifier. Furthermore, we hypothesize that because the UKYSmall dataset was larger than CMC, we were able to take advantage of tf-idf weighting and used bigrams, unigrams, and CUIs as features. For the features used for the models in Table 2.2, we used feature selection with BNS and also removed features that did not occur at least 5 times in the training set. Overall, we show the results for four combinations: 1. BR with LR; 2. BR with OTS and probabilistic thresholding (MLPTO); 3. BR with optimal training sets (OTS) and always picking 3 labels (RCut 3); and 4. Copy Transformation (RCut 3). The second row, the combination of OTS, BNS, and MLPTO gives the best F-scores across all three types of measures although other combinations in rows 3 and 4 offer a higher recall with substantial losses in precision. Since these methods use RCut 3, that is, three labels are always picked for each EMR, we can see how the recall gain also introduced many false positives, a scenario that MLPTO seems to have handled effectively with separate thresholding for each instance. Finally, we can see from the first row, where only BR is used without any other components, diagnosis code extraction is a difficult problem that does not lend itself to simple basic transformation approaches in multi-label classification.

In Table 2.3, we show the results of learning component ablation on our best classifier (from row 2 of Table 2.2) shown as the first row. While removal of MLPTO and OTS caused losses of up to 8% in F-scores, dropping feature selection with BNS caused a drop of 30% in F-score, which clearly demonstrates the importance of appropriate feature selection in these combinations. However, interestingly, just using BNS alone without MLPTO and OTS did not result in major performance gains compared to the first row of Table 2.2.

2.7.4 Conclusion

In this section, we described preliminary work on two small medical datasets, CMC and UKYSmall. Moreover, we show traditional methods such as linear models, feature section, and under-sampling perform well on both datasets. This section contrasts with the methods we describe in future chapters. Specifically, in this section, the methods depend on hand-crafted features (feature engineering), whereas future sections focus more on architecture engineering and modifying neural network training procedures. A few of the components described in this section will not easily scale to tens of thousands of labels which are available in the ICD terminologies. For example, performing feature selection and data subset selection (OTS) for each label can easily become infeasible as the number of features and number of labels grow. In large datasets, there can be more than a million ngrams (unigrams and bigrams).

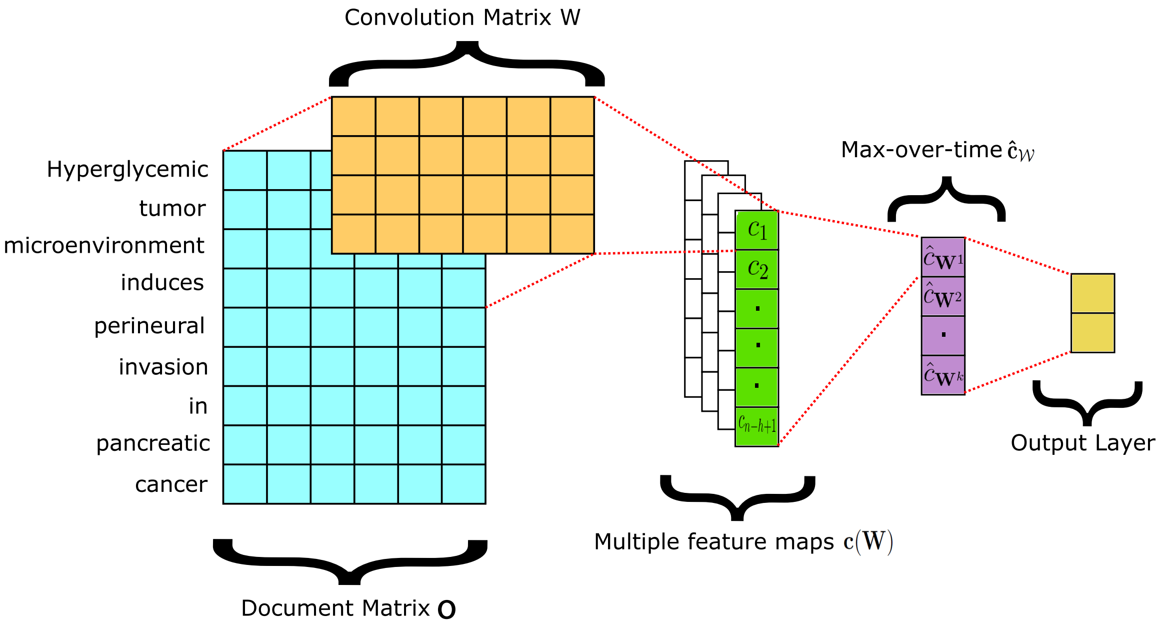
While there are many differences between the methods described in this section and the methods in future chapters, there are also commonalities in terms of the problems they try to solve. For example, in this section, we describe both the MLPTO and RCut thresholding strategies. We see even with small label sets, thresholding is an important factor for certain evaluation measures. This issue is amplified as the total number of labels grows. We discuss this further in Chapter 4.

In summary, for these small datasets, we did not encounter the two major issues that plague real-word datasets. For example, the CMC dataset contains only 21 words per example on average. Contrast this with Chapter 3 which contains EMRs with over 10000 words. Likewise, with only 56 codes in the UKYSmall dataset, and 45 labels in the CMC dataset, we did not have to address infrequent codes that may occur only once in the training dataset, or worse, never appear in the training dataset at all. Both of these issues are addressed in future chapters.

2.8 Convolutional Neural Networks for Text Classification

Convolutional neural networks (CNNs) are a substantial part of the methods introduced in Chapters 3, 4, and 5. In this section, we provide background on CNNs for text classification. Specifically, we describe the model we applied to a biomedical text classification problem in Rios and Kavuluru (2015b). We also provide an intuitive

Figure 2.1: Traditional CNN model for text classification which consists of three main components: A convolutional layer, max-over-time pooling, and a final output layer.



explanation of CNNs, and subsequently elaborate with a more detailed specification of the model and the standard training process.

The central concept in CNNs is the notion of *convolution filters* (CF) that are traditionally used in signal processing. The general principle is to learn several CFs which are able to extract useful features from a document for the specific classification task based on the training dataset. This has proven to be very useful in computer vision (Zeiler et al., 2010) where convolutions learn high level features of an image (e.g., curves and faces). Before we go into the specifics, we provide a high-level overview of convolutions for text classification.

2.8.1 Convolutions and CFs

A convolution is a binary operation involving the following operands: a segment of text and a specific CF, both of which are represented as real matrices for our purposes. The output is a single real number. The matrix representation of the text segment, which is typically a contiguous sequence of words in the document, is composed of the word vectors of tokens that constitute it. A CF is also a matrix of the same dimensions as the text segment matrix. A specific CF operates on all contiguous segments of a document using a sliding window, producing as many real number outputs as there are contiguous segments of a certain length in the document. This sequence of real numbers is called the feature map associated with the particular CF being used. Different CFs produce different feature maps, which can then be used as features in text classification. The overview of the convolution operation and CFs explained here forms the convolutional layer of the CNN. Traditionally, there is a conventional softmax layer that takes as input the feature maps and outputs class probability estimates. The main idea is to learn CFs (that is, the elements in the corresponding matrices) that produce better feature maps that optimize our objective function. The learning process is based on predicting classes for training instances and making adjustments to the CF elements through back propagation of the gradients of the objective function (conditional log-likelihood of training data) being optimized. In this intuitive explanation, we left out many details including the mathematical definition of the convolution operation, the objective function, regularization (to deal with overfitting), and the stopping protocol for the learning process. The following section discusses these in detail.

2.8.2 Model Specification

The architecture of the full CNN model described in this section is shown in Figure 2.1. It has two layers, including a single convolution layer, and a fully connected output (softmax) layer.

The base component in the model is a word vector $\mathbf{w} \in \mathbb{R}^d$, where d is the dimension of the word vectors. A document is represented as a matrix $\mathbf{O} \in \mathbb{R}^{n \times d}$, where n is the number of words in it and each row represents the word vector for the corresponding token. To simplify the equations in this section we will assume the ground truth for the document $Y \in \mathbb{R}^2$ 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 is more aligned with the two output nodes of the final layer for the two corresponding classes (positive/negative) for each binary classifier. Although a single node would have been sufficient for binary classification, it is also possible to chose to build a model with multiple nodes to simply have the code set-up for multi-class classification for other text classification problems.

We define a CF $\mathbf{W} \in \mathbb{R}^{h \times d}$, where h is the number of words we wish the convolution filter to span, that is, the length of the sliding window. Let the 2-D convolution operation $*$ be defined as

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

We next map a length h word window, $\mathbf{O}_{j:j+h-1}$, of the document to a real number $c_j \in \mathbb{R}$ using a non-linear function f as

$$c_j = ReLU(\mathbf{W} * \mathbf{O}_{i,j:j+h-1} + b),$$

where $b \in \mathbb{R}$ is the bias term for the convolution filter, and $ReLU$ represents the rectified linear function (Glorot et al., 2011; Nair and Hinton, 2010). After convolving over the entire document using $\mathbf{W}$, we have the corresponding convolved feature map

$$\mathbf{c}(\mathbf{W}) = [c_1, c_2, \ldots, c_{n-h+1}].$$

To overcome the issue of varying document lengths we perform max-pooling (Collobert et al., 2011) operation

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

which gives a single feature $\hat{c}_{\mathbf{W}}$ corresponding to the feature map generated by $\mathbf{W}$. However, we will learn several CFs, say k of them, $\mathbf{W}^1, \ldots, \mathbf{W}^k$, to create multiple feature maps leading to the corresponding single max-pooled features $\hat{c}_{\mathbf{W}^t}$, $t = 1, \ldots, k$. These form a final max-pooled feature vector

$$\hat{\mathbf{c}}_{\mathcal{W}} = [\hat{c}_{\mathbf{W}^1}, \ldots, \hat{c}_{\mathbf{W}^k}]^T, \tag{2.7}$$

where $\mathcal{W} = \{\mathbf{W}^1, \ldots, \mathbf{W}^k\}$.

After obtaining $\hat{\mathbf{c}}_{\mathcal{W}}$, we add a final softmax layer. Let $\mathbf{U} \in \mathbb{R}^{2\times k}$ and $b^U \in \mathbb{R}^2$ be the parameters of the softmax layer with weighted inputs

$$s_j = \mathbf{U}_j \hat{\mathbf{c}}_{\mathcal{W}} + b_j^U \tag{2.8}$$

and output label probability estimates

$$\hat{y}_j = \frac{e^{s_j}}{\sum_i e^{s_i}}, \tag{2.9}$$

where $\mathbf{U}_j$ is the j-th row of $\mathbf{U}$, b_j^U is the j-th element of b^U, and y_j is the j-th label for the document corresponding to matrix $\mathbf{O}$.

If $\mathcal{D}$ is the set of training document matrices, to learn each classifier we minimize

$$-\sum_{D \in \mathcal{D}} \log(\hat{y}_{pos}^{D}), \tag{2.10}$$

where $pos = 1$ ($pos = 2$) if the corresponding document is a negative (positive) instance for document D. The parameters of the CNN ($\mathcal{W}, b, \mathbf{U}, b^U$ and the word vectors) that minimize this are obtained by calculating the gradient and using back propagation with the stochastic gradient descent approach. A subtle but crucial aspect that makes CNNs for NLP tasks different from those used in computer vision is that the base input components to the CNN, that is, the word vectors, are also modified using back propagation in addition to the traditional network weights. This is done by treating the word vector elements as network weights of the first and so called projection layer (Socher, 2014, Chapter 2) and modifying them just like any other network weight. However, the vector elements for a given word only change when the current instance contains that word, which happens often if it is a common word or if the dataset is large. We used the popular mini-batches (Ngiam et al., 2011) approach instead of updating parameters for each example. CNNs also warrant multiple epochs where the learning process goes through the entire training dataset multiple times in optimizing equation 2.10.

Instead of using the well known l1 or l2 regularization CNNs can be regularized by using the dropout (Srivastava et al., 2014) option to prevent over-fitting during training. Specifically, instead of passing s_j (from equation 2.8) to the softmax function in equation 2.9 during training, we actually pass

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

where $\circ$ refers to element-wise multiplication and $\mathbf{r} \in \{0,1\}^k$ is constructed with each r_i drawn from the Bernoulli distribution with parameter p (typically set to 0.5). Intuitively, this means that gradients are backpropagated only through unmasked elements where $r_i = 1$. During test time we scale the weights U such that

$$s_j = p(\mathbf{U}_j \hat{\mathbf{c}}_{\mathcal{W}} + b_j^U).$$

This down weighting is essential since at training, on average, only half of the U edges were active, which is not true at test time. Besides dropout, early-stopping is also essential in order to help combat over-fitting. Typically, early-stopping is done by simply terminating the training of the model when the desired score on a held out validation dataset does not increase in performance. However, we find that this typically causes our model to stop training too early. To overcome this issue, training can be stopped if there were 5 consecutive epochs in the training procedure that did not increase the validation score. For example, if there was an increase in F-score after the second epoch on the validation dataset, and training continued for five more epochs without any further increase, we kept the model from the 2nd epoch. Another option is to train for a fixed number of epochs and epoch after each epoch. The epoch which provides the highest validation F-score.

2.8.3 Discussion

In this section, we described a popular CNN architecture used for text classification (Kim, 2014; Rios and Kavuluru, 2015b). In future Chapters, we use the basic components of this CNN and describe three extensions that answers one or more of the following questions: How should we handle multi-label datasets rather than multi-class? Is there a better architecture that better handles the extreme label imbalance experienced in EMR datasets? How do we handle classes that never occurred in the training dataset?

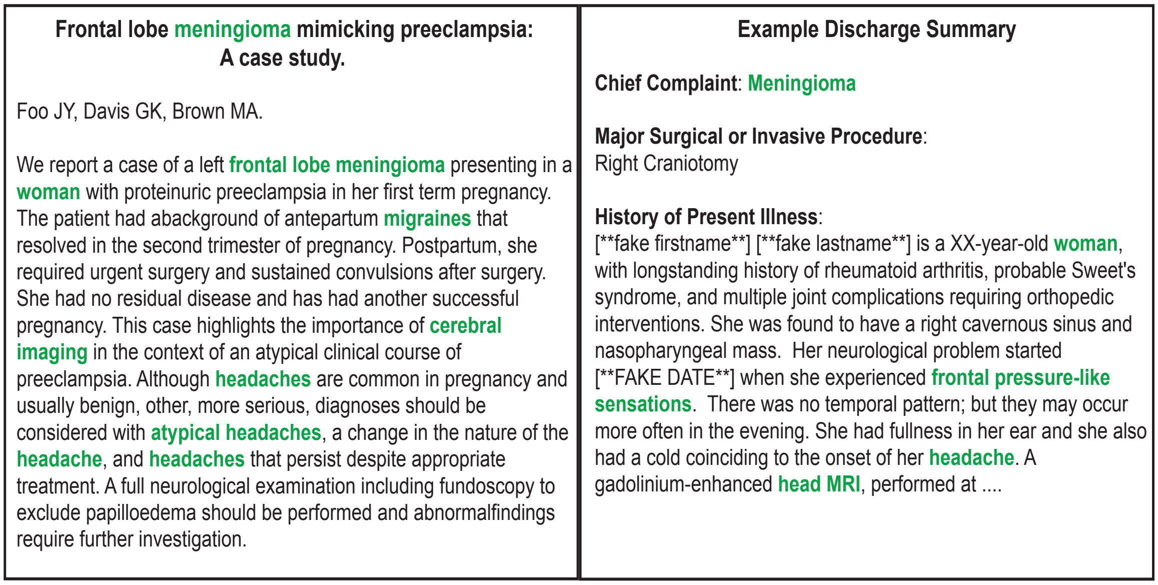
Chapter 3 Extracting Diagnosis Codes using Transfer Learning from EMRs

As stated in Chapter 1, medical coding datasets are often plagued with the problem of data sparsity. EMR datasets may contain tens of thousands of records. However, given the large number of diagnosis and procedure codes, only a few training examples may be available for each code. It is also common for many diagnosis and procedure codes to never appear in the training dataset. In this chapter, we introduce a transfer learning training methodology which improves the performance of CNNs on both frequently and semi-infrequent occurring codes. The method described in this chapter does not handle the extreme tail labels in the dataset — labels that occur only a few times or labels that never appear in the training dataset. However, we show that transfer learning can substantially improve labels that occur frequently enough for traditional supervised learning techniques.

Much of the prior work on automated ICD coding has trained models from scratch, which means the models assume zero prior knowledge about the domain. However, expert domain knowledge is abundant for various medical applications. If we want to build models that can predict infrequent labels, then it is essential to take advantage of all available information we have about the problem. Some of the available knowledge sources are in structured form. For example, the Unified Medical Language System (UMLS) (Bodenreider, 2004) is a comprehensive thesaurus and ontology of biomedical concepts. However, much of the available information is in the form of unstructured text. Medline indexes more than 27 million biomedical research articles. The articles are available to the general research community via the PubMed search engine. Some of the research articles contain relevant information about treating specific diseases or illnesses. Moreover, many of the indexed articles are "case reports" which describe the symptoms, diagnosis, and treatment of individual patients. We show an example abstract indexed by Medline in Figure 3.1a, and an example discharge report in Figure 3.1b. If we compare the abstract to the "History of Present Illness" section in Figure 3.1b, then we can see how this auxiliary data may be useful. For example, in the figure we observe that the patient experienced atypical headaches which should have been a sign of a serious illness (i.e., meningioma). Likewise, the EMR in Figure 3.1b also reports headaches as a symptom.

How can we use Medline indexed articles to improve ICD-9 code prediction? State-of-the-art results have been achieved in text classification using CNNs with neural

Figure 3.1: In Figure 3.1a, we show an example title and abstract from the Medline indexed paper by Foo et al. (2017). In Figure 3.1b, we show an example snippet from a discharge summary in the MIMIC III dataset (Johnson et al., 2016).



Frontal lobe meningioma mimicking preeclampsia:
A case study.

Foo JY, Davis GK, Brown MA.

We report a case of a left **frontal lobe meningioma** presenting in a **woman** with proteinuric preeclampsia in her first term pregnancy. The patient had abackground of antepartum **migraines** that resolved in the second trimester of pregnancy. Postpartum, she required urgent surgery and sustained convulsions after surgery. She had no residual disease and has had another successful pregnancy. This case highlights the importance of **cerebral imaging** in the context of an atypical clinical course of preeclampsia. Although **headaches** are common in pregnancy and usually benign, other, more serious, diagnoses should be considered with **atypical headaches**, a change in the nature of the **headache**, and **headaches** that persist despite appropriate treatment. A full neurological examination including fundoscopy to exclude papilloedema should be performed and abnormalfindings require further investigation.

Example Discharge Summary

Chief Complaint: **Meningioma**

Major Surgical or Invasive Procedure:
Right Craniotomy

History of Present Illness:
[**fake firstname**] [**fake lastname**] is a XX-year-old **woman**, with longstanding history of rheumatoid arthritis, probable Sweet's syndrome, and multiple joint complications requiring orthopedic interventions. She was found to have a right cavernous sinus and nasopharyngeal mass. Her neurological problem started [**FAKE DATE**] when she experienced **frontal pressure-like sensations**. There was no temporal pattern; but they may occur more often in the evening. She had fullness in her ear and she also had a cold coinciding to the onset of her **headache**. A gadolinium-enhanced **head MRI**, performed at ....

(a) (b)

word embeddings. However, traditional CNN models require a large amount of training data, and using them for multi-label datasets becomes problematic for large label spaces because many labels occur infrequently. To overcome this issue, we use *transfer learning* (Oquab et al., 2014) to take advantage of the biomedical articles indexed by Medline. Each article indexed by Medline is annotated with a set of Medical Subject Headings (MeSH). For example, there is a specific MeSH term for "meningioma" (D008579). In this case, there is a 1-to-many match to the ICD-10-CM codes C70.0 and D32.0. Given the textual similarities observed in Figure 3.1, if we pass an EMR in Figure 3.1b to a model trained to predict MeSH terms, then the model may predict D008579. Transfer learning is a machine learning technique which improves the predictive performance on a new task by transferring knowledge from a different but related task. We use transfer learning to improve the performance of automated medical coding systems (target task) by "transferring" knowledge acquired from learning to predict Medical Subject Headings (MeSH) for biomedical articles indexed on Medline (source task). Intuitively, instead of forcing our model to learn how to represent documents for diagnosis code prediction with a limited dataset, we pretrain a CNN on a larger dataset of Medline abstracts to compute intermediate document representations useful for assigning diagnosis codes to EMRs. Furthermore, we introduce a simple, yet effective, method to fine-tune the document representations to the target task without forgetting the information learned from the source task.

Overall, the goal of this chapter is to study the effect of transfer learning for ICD code prediction. We want to answer the following questions: Can transfer learning improve CNNs for medical coding? If transfer learning helps, then what is the best transfer learning method which achieves the largest increase in performance?

We summarize the contributions of this chapter below:

- Our method uses CNNs (Kim, 2014; Baumel et al., 2017) to efficiently train on examples with more than 5000 words. We also propose a simple, yet effective, transfer learning method to improve the performance of CNNs for ICD classification without forgetting information learned on the source task.
- We provide a comprehensive analysis comparing our method with our prior work on extracting diagnosis codes. Furthermore, besides our proposed transfer learning approach, we compare several other transfer learning methodologies to understand what works best for our dataset.

The rest of this chapter is organized as follows: In Section 3.1 we discuss related work about transfer learning. Section 3.2 presents the dataset used for our study and discusses the various transfer learning methods we use in our experiments. Next, in Section 3.3 we compare our method to prior work and present a detailed analysis of different transfer learning approaches. Finally, in Section 3.4 we summarize our contributions presented in this chapter.

3.1 Related Work: Transfer Learning

Like neural networks, transfer learning has shown impressive improvements in classification applications for computer vision. Oquab et al. (2014) show that parts of neural networks trained on large datasets can be used to generate features for datasets with a small number of training examples. More recently, Mou et al. (2016) explored the application of transfer learning to NLP tasks. In a similar manner to Oquab et al. (2014), Mou et al. (2016) show that transfered neural network features are useful for prediction. Al-Stouhi and Reddy (2016) show that transfer learning can improve classification performance in the presence of label imbalance. This result is promising given EMR power-law datasets generally contain large imbalances between different ICD codes.

Our method addresses similar concepts as Mou et al. (2016), where they study how to apply transfer learning to various NLP tasks to understand two questions. First, does transfer learning help in NLP? Second, what is the best way to implement transfer learning? Besides the different application domains, we also introduce a different transfer learning method not explored by Mou et al. (2016). Al-Stouhi and Reddy (2016) also emphasizes the use of transfer learning-like methods to improve problems with label imbalance. Compared to our work, Al-Stouhi and Reddy (2016) do not take advantage of recent advances in neural networks, and instead use a boosting-based classifier. Howard and Ruder (2018) show that transfer learning approaches produce significant improvements by training only a language model on the source domains. This is a useful method for taking advantage of unlabeled data.

Recently, transfer learning has been shown to be useful in biomedical research. Wiens et al. (2014) discuss issues with model generalization across different hospitals. For example, different hospitals may have different norms, or even EMR structures, which cause models to perform poorly if the model is trained on data from a different hospital. Choi et al. (2016) use transfer learning to improve model performance across different hospitals; however they model disease progression rather than performing text classification. Transfer learning has also been shown to improve biomedical relation extraction Sahu and Anand (2017). Sometimes the target data has the same classes as the source dataset. However, the data distributions vary in such a way that training on the source dataset does not generalize well. In Rios et al. (2018), we propose a transfer learning-like technique using domain adaptation for biomedical relation extraction. This method assumes no labeled training data is available for the target dataset.

3.2 Materials and Methods

Our dataset contains 71,463 EMRs and a total of 7,485 unique diagnosis codes based on in-patient visits to the University of Kentucky (UKYLarge) hospital between 2011 to 2012. Each EMR is annotated with a set of ICD-9 codes[1]. Following our prior work (Kavuluru et al., 2015), and similar to Chapter 2.7, we preprocess our data by truncating all ICD-9 codes of the form *abc.xy* to *abc.x* and we remove all codes that occur in less than 50 EMRs. Truncating and removing the most infrequent ICD-9 diagnosis codes results in a total of 1,231 codes which we use for classification.

---

[1] We realize that US health care facilities have moved to ICD-10-CM as of Oct 1, 2015. Given this is a much recent move, it has limited the availability of training data with ICD-10 codes. Hence as proof of concept for transfer learning, we experimented with ICD-9 codes.

Figure 3.2: ICD-9 code frequency distribution of the UKYLarge EMR dataset.

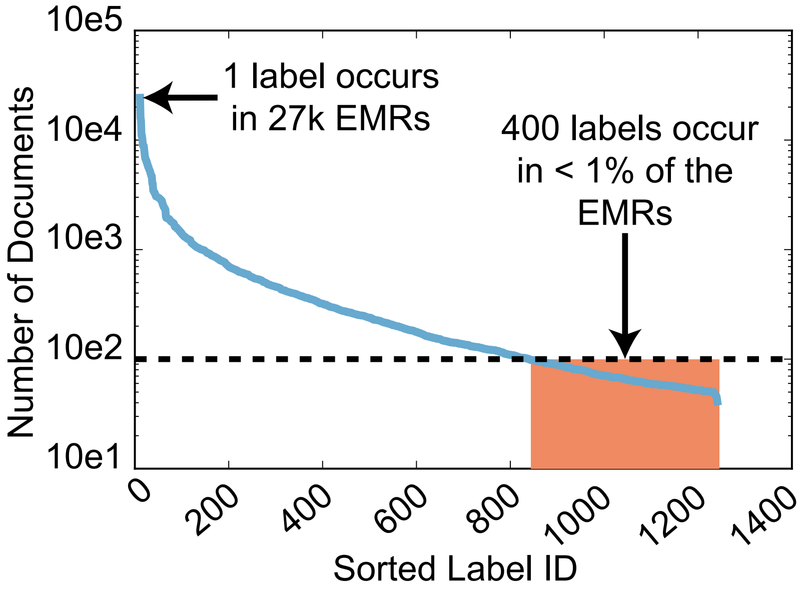
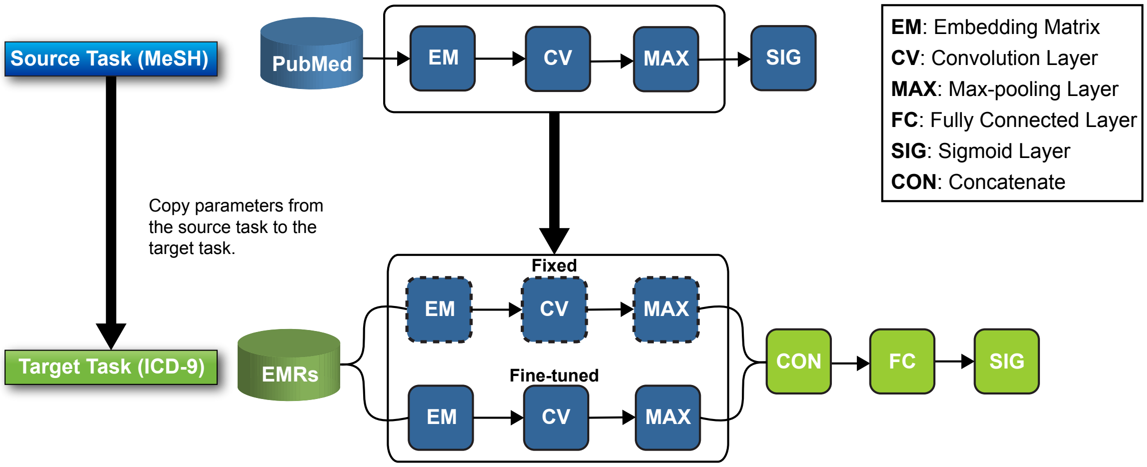


Table 3.1: Transfer learning dataset statistics.

	Medline	UKLarge
# Instances	1,600,000	71,463
# Labels	27,150	1,231
Label Cardinality	12.62	7.4
Avg # Words per Instance	147	5,303
# Code Combinations	–	60,238

Intuitively, by truncating the labels and removing codes that occur infrequently, we reduce the extreme tail of the frequency distribution — codes that only occur a few times in the dataset. Traditional NN-based methods will not be able to predict labels that occur only a few times, even with transfer learning. We explore the extreme tail labels in more detail in Chapter 5. From the full dataset, 2000 EMRs are randomly removed to create a validation dataset and 3000 EMRs are held-out for final testing; the remaining EMRs (over 65,000) are used for training. The frequency distribution over all codes is available in Figure 3.2. We find that 1 diagnosis code occurs in more than 27k EMRs which is nearly 38% of the entire dataset. 400 diagnosis codes occur in no more than 100 EMRs. Basic statistics about the datasets are shown in Table 3.1.

Figure 3.3: The parameters learned in the source task are transferred to the target task model and fixed while the target task specific model parameters are updated during training.



3.2.1 Overview

Figure 3.3 provides a high-level overview of our method. Transfer learning involves training two models, one model for a source task (stage 1) and the other for a target task (stage 2). Each model is trained on a different dataset. For the source task (stage 1), we collected 1.6 million Medline citations (title and abstracts) and trained a CNN model to predict MeSH terms. All of the parameters of the source task model, except for the output layer, are used to initialize the parameters for the target task model. Finally, for stage 2, the target model is trained on the UKYLarge EMR dataset to predict ICD codes.

3.2.2 Convolutional Neural Networks for Text Classification

We use the CNN model described in Chapter 2 as our base model to represent each EMR. Specifically, we use the max-pooled features defined by Equation 2.7. For simplicity we refer to this vector as

$$g(\mathbf{x}) = \hat{\mathbf{c}}_{\mathcal{W}}$$

where $g(\mathbf{x}) \in \mathbb{R}^k$. k is the number of convolution filter widths times the number filters trained for each width. Furthermore, the remainder of this chapter will refer to the convolution filters $\mathcal{W}$ which form the convolution layer as "CV" and the embedding layer (all the word vectors) is referred to as "EM".

3.2.3 Stage 1: Training on Source

To use transfer learning techniques, we first train our model on the source data. Given $g(\mathbf{x})$, we pass it through multiple sigmoid outputs, one for each label

$$\hat{\mathbf{y}}_{\mathcal{S}} = sigmoid(\mathbf{W}_{\mathcal{S}}\, g(\mathbf{x}) + \mathbf{b}_{\mathcal{S}})$$

where $\mathbf{W}_{\mathcal{S}} \in \mathbb{R}^{L_{\mathcal{S}} \times k}$, $\mathbf{b}_{\mathcal{S}} \in \mathbb{R}^{L_{\mathcal{S}}}$, and $L_{\mathcal{S}}$ is number of source labels. The sigmoid function is defined as

$$sigmoid(x) = \frac{1}{1 + e^{-x}}.$$

For multi-label classification, sigmoid units are required rather than the softmax layer used for multi-class classification. Each element, $\hat{\mathbf{y}}_i$, produces a score for each label using the sigmoid squashing function which constrains the score to the range $[0, 1]$.

We train over all labels jointly by minimizing the multi-label binary cross-entropy loss (Nam et al., 2014a)

$$\mathcal{L} = -\sum_{l}^{L_{\mathcal{S}}} y_l log(\hat{y}_l) + (1 - y_l) log(1 - \hat{y}_l).$$

where L_s is the number of labels in the source task. The loss, J_{CE}, can be optimized using stochastic gradient descent (SGD).

3.2.4 Stage 2: Transfer Learning

We experiment with three traditional transfer learning approaches and introduce a new method. The three traditional methods take the model trained on the source dataset and replace the output layer with two additional layers. First, given $g(\mathbf{x})$, the max-pooled feature vector, we pass it through a fully-connected layer

$$\mathbf{h} = ReLU(\mathbf{W}_a\, g(\mathbf{x}) + \mathbf{b}_a) \tag{3.1}$$

where $\mathbf{W}_a \in \mathbb{R}^{a \times k}$ and $\mathbf{b}_a \in \mathbb{R}^{a}$. In transfer learning literature, this layer is known as as an "adaptation layer" (Oquab et al., 2014). The adaptation layer learns to transform the mid-level features optimized on the source dataset to better represent the target data.

Next, $\mathbf{h}$ is passed to a target specific output layer

$$\hat{\mathbf{y}} = sigmoid(\mathbf{W}_o\, \mathbf{h} + \mathbf{b}_o) \tag{3.2}$$

where $\mathbf{W}_o \in \mathbb{R}^{L \times a}$, $\mathbf{b}_o \in \mathbb{R}^{L}$, and L is the total number of target labels.

As previously stated, we experiment with three recently proposed transfer learning methods. Each method shares the same overall CNN architecture. However, they vary based on which parameters are updated while training on the target dataset. We describe the different variations below:

- EM[✗] CV[✗] – For this variation, all parameters used during stage 1 including the word vectors EM, and convolution weights CV are not updated during the stage 2 training process. However, The adaptation layer parameters, $\mathbf{W}_a$ and $\mathbf{b}_a$, and the target output layer parameters, $\mathbf{W}_o$ and $\mathbf{b}_o$, are updated.
- EM[✗] CV[✓] – This method is initialized with the CNN weights after Stage 1. Similar to the previous method, we keep the word embeddings fixed. However, the convolution parameters CV are fine-tuned during stage 2.
- EM[✓] CV[✓] – The third method expands on EM[✗] CV[✓] by fine-tuning both the word embeddings and convolution parameters while training on the target dataset.

We also introduce a simple, yet effective, transfer learning method. Transfer learning methods that fine-tune the weights transferred from the source task tend to forget what they have learned from the source dataset (Li and Hoiem, 2017; Kirkpatrick et al., 2017). Generally, this issue is measured by testing how well the fine-tuned NNs perform on the original source task after fine-tuning. In our case, we are only concerned about predictive performance on the target task, assigning ICD diagnosis codes to EMRs. Therefore, we do not care how well our model performs on the source dataset. However, we hypothesize if we forget information about the source dataset, our model will not generalize as well. We believe this given the high-level of similarity between the two domains. To overcome the issue of catastrophic forgetting, we propose the method EM[✓] CV[✓] + EM[✗] CV[✗]. Specifically, we make two copies of the word embeddings and convolution parameters learned during stage 1. The two copies are used to generate two mid-level representations of each document, $g(\mathbf{x})$ and $g'(\mathbf{x})$. Both representations are concatenated

$$\mathbf{h}^2 = g(\mathbf{x}) \mid\mid g'(\mathbf{x})$$

where $\mathbf{h}^2 \in \mathbb{R}^{2k}$. $\mathbf{h}^2$ is then passed to the adaptation layer defined in Equation 3.1, then to the output layer defined by Equation 3.2. During training, we only optimize the word embedding and convolution parameters used to generate $g(\mathbf{x})$. The parameters that create $g'(\mathbf{x})$ are not updated.

3.2.5 Word Dropout

EMRs in the UKYLarge dataset contain more than 5000 words per instance on average. A few examples in the dataset contain more than 10000 words. Training on lengthy instances can take a long time and uses a lot of memory on the GPU. To improve training efficiency, we use word dropout (Iyyer et al., 2015). Similar to dropout which randomly sets some unit weights to zero to avoid overfitting, during training, word dropout completely removes words from an EMR at random. Besides reducing the overall training time, word dropout also reduces overfitting. Intuitively, by randomly removing words from document, we artificially create new documents. For long documents, we assume that removing words from the document will not substantially change the overall meaning of what the EMRs describe.

3.2.6 Ensemble

It is possible for our model to overfit to infrequently occurring labels. Wallace et al. (2011) show that bagging multiple oversampled classifiers improves the performance of infrequent labels in the multi-class setting. However, oversampling is not trivial in the multi-label setting. Averaging multiple NNs trained with different seeds is a well known way to improve performance (Huang et al., 2017) of NNs in general. Therefore, we train Γ different models, each initialized with a different random seed. At test time, the predictions for each model are averaged

$$\hat{\mathbf{y}}_e = \frac{1}{\Gamma} \sum_{i=1}^{\Gamma} \hat{\mathbf{y}}^i$$

where $\hat{\mathbf{y}}_e \in \mathbb{R}^L$ and $\hat{\mathbf{y}}^i$ represents the predictions for the i-th model.

3.3 Experiments

In this section, we compare our work with prior medical coding methods on the UKYLarge dataset. We also analyze how our transfer learning model compares to related methods.

3.3.1 Implementation Details

Our model uses convolution filter widths that span 3, 4, and 5 words. We train 300 filters for each filter width. Therefore, the size of the max-pooled feature vectors $g(\mathbf{x})$ will have a dimensionality of 900. The word embedding dimensionality is set 300. We use standard dropout before the final output layer with a dropout probability of 0.5. The dropout probability for word dropout is set to 0.3. Furthermore, we truncate all documents to a max length of 6000 words. The model is optimized using the SGD variant AdaDelta (Zeiler, 2012) with a learning learning rate of 0.001 and a minibatch size of 50.

3.3.2 Baseline Methods

We compare against three different methods:

1. Logistic Regression (LR) trained on tf-idf weighted n-grams,
2. LR + L2R + NERC which uses label scores from LR, the k-nearest neighbor similarity scores, and named entity recognition based codes (NERC) extracted using NLM's MetaMap as features to a second-level stacking-like learning-to-rank (L2R) method,
3. and a model averaging ensemble with three CNNs without transfer learning.

We also compare two versions of each transfer learning method, an ensemble model that averages 3 models trained with different seeds, and a single model with out model averaging. We evaluate our method using the popular micro F-score measure. Because we are not only interested in predicting frequent ICD codes, we also use macro F-score which gives equal weight to all labels independent of the label frequency. Both of these metrics are defined in Chapter 2.6.

3.3.3 Layer by Layer Analysis

In Table 3.2 we compare the different transfer learning variations. We find that updating parameters always outperforms transfer learning methods that keep parameters fixed. For example, EM[✗] CV[✓] outperforms EM[✗] CV[✗] by more than 3% with respect to micro F-Score. Likewise, EM[✓] CV[✓] improves by more than 3% over the micro F-Score obtained by EM[✗] CV[✓]. Without ensembling, we find only a small improvement in micro F-Score using the EM[✓] CV[✓] + EM[✗] CV[✗] method.

Table 3.2: Layer-by-layer results for various transfer learning methodologies.

	Micro F-Score	Macro F-Score
EM[✗] CV[✗]	46.5	23.6
EM[✗] CV[✓]	49.8	23.8
EM[✓] CV[✓]	53.1	24.2
EM[✓] CV[✓] + EM[✗] CV[✗]	53.5	26.0
EM[✗] CV[✗] AVG	48.3	25.5
EM[✗] CV[✓] AVG	51.3	25.5
EM[✓] CV[✓] AVG	54.1	25.8
EM[✓] CV[✓] + EM[✗] CV[✗] AVG	**56.7**	**28.6**

Table 3.3: Results comparing conventional approaches, CNNs, and CNNs with transfer learning

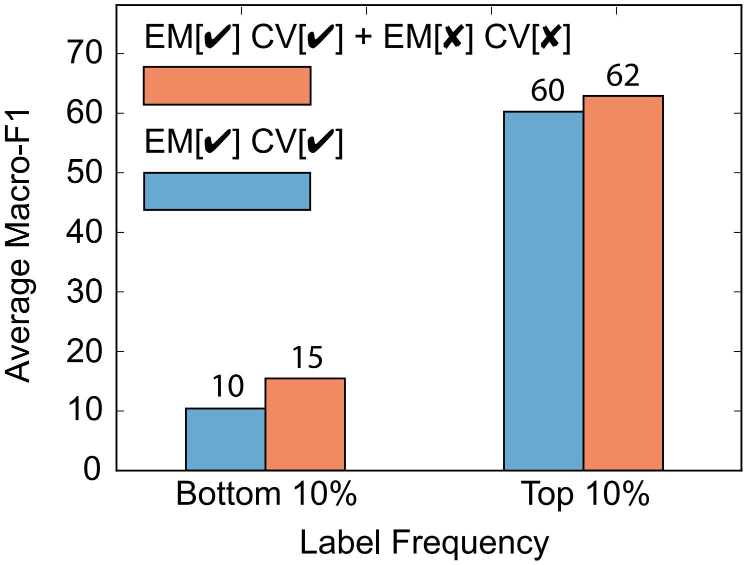
	Micro F-Score	%-increase	Macro F-Score	%-increase
LR (Kavuluru et al., 2015)	48.2	–	19.8	–
LR + L2R (Kavuluru et al., 2015)	49.5	1.3%	21.2	1.4%
LR + L2R + NERC (Kavuluru et al., 2015)	49.9	1.7%	23.0	3.2%
EM[✓] CV[✓] + EM[✗] CV[✗]	53.5	5.3%	26.0	6.2%
EM[✓] CV[✓] + EM[✗] CV[✗] AVG	**56.7**	**8.5%**	**28.6**	**8.8%**

However, we find nearly a 2% improvement with respect to macro F-score. If updating all the parameters outperforms methods which fixes the weights, then does this imply that catastrophic forgetting is not an issue for our task? The difference in macro F-score between EM[✓] CV[✓] and EM[✗] CV[✗] is only 0.6%. Yet, EM[✓] CV[✓] + EM[✗] CV[✗] improves the macro F-score over EM[✓] CV[✓] by nearly 2%. This result suggests that forgetting source task information may not negatively impact infrequent codes when we update the parameters on the target task. However, it also does not improve the performance of infrequent codes either. When we update the weights and store an extra copy of the source copy paramters (EM[✓] CV[✓] + EM[✗] CV[✗]), then it generalizes better across all ICD-9 diagnosis codes regardless of the code frequency in the dataset.

3.3.4 Comparison with prior work

In Table 3.3 we compare our proposed transfer learning method with prior work on the UKYLarge EMR dataset. We improve over LR by more than 8% for both the micro and macro F-Scores. Our ensemble method improves on "LR + L2R + NERC" by

Figure 3.4: This figure compares the macro F-scores on the 10% least frequent codes to the macro F-score on the 10% most frequent ICD-9 diagnosis codes.



nearly 7% micro F-Score which suggests that NNs can better predict frequent labels. Likewise, the ensemble approach improves on the prior best macro F-Score by more than 5%. Even without ensembling, we improve over LR + L2R + NERC by 3% with respect to the macro F-score. Overall, we find that even in the presence of data sparsity, NNs can outperform traditional text classification methods when we use transfer learning.

3.3.5 Label Frequency Analysis

In Figure 3.4 we analyze the macro F-Scores of the 10% least frequent and 10% most frequent diagnosis codes in the UKYLarge dataset. While calculating the macro F-score over all labels gives some insight about how our method performs on infrequent labels, if the frequent and infrequent codes are jointly compared, then it confounds its interpretation. We find that our proposed method improves infrequent label performance by 5%. The macro-averaged performance improves by 2% for frequent classes. Compared to EM[✓] CV[✓], these results suggests that the source information EM[✓] CV[✓] + EM[✗] CV[✗] avoids forgetting has a greater impact on infrequent codes.

3.4 Conclusion

In this chapter, we demonstrated the potential of transfer learning using CNNs for biomedical text classification over conventional CNNs, and other traditional ensemble approaches. For comparative purposes, we restricted our dataset to labels occurring at least 50 times after preprocessing. In this setting most neural networks can reliably be trained to predict all codes. However, even though this section used larger label spaces compared to Chapter 2.7, this reduced label space is not realistic. In Chapter 5 we develop a method which can handle the full label set — including codes occurring much less than 50 times.

Chapter 4 EMR Coding with Semi-Parametric Multi-Head Matching Networks

Are there neural architectures that can better handle the label distributions (Figure 1.2) encountered in EMR datasets? Recent advances in *extreme multi-label classification* have proven to work well for large label spaces. Many of these methods (Yu et al., 2014; Bhatia et al., 2015; Liu et al., 2017) focus on creating efficient multi-label models that can handle 10^4 to 10^6 labels. While these models perform well in large label spaces, they don't necessarily focus on improving prediction of infrequent labels. Typically, they optimize for the top 1, 3, or 5 ranked labels by focusing on the P@1, P@3, and P@5 evaluation measures. The labels ranked at the top usually occur frequently in the dataset and it is not obvious how to handle infrequent labels. One solution would be to ignore the rare labels. However, when the majority of medical codes are infrequent, this solution is unsatisfactory. In Chapter 3, we improve traditional CNNs by taking advantage of external biomedical textual data. Yet, we ignore labels which occur less than 50 times in the training dataset. In this chapter we develop a novel neural network which can better handle infrequent labels. Furthermore, we analyze various neural-based methods across the entire spectrum of labels independent of label frequency.

While neural networks have shown great promise for text classification (Kim, 2014; Yang et al., 2016; Johnson and Zhang, 2017), the label imbalances associated with EMR coding hinder their performance. In Chapter 3, we focused on codes that occurred at least 50 times in the training dataset. Imagine if a dataset contains only one training example for every class leading to *one-shot learning*, a subtask of *few-shot learning*. How can we classify a new instance? A trivial solution would be to use a non-parametric 1-NN (1 nearest neighbor) classifier. 1-NN does not require learning any label specific parameters and we only need to define features to represent our data and a distance metric. Unfortunately, defining good features and picking the best distance metric is nontrivial. Instead of manually defining the feature set and distance metric, neural network training procedures have been developed to learn them automatically (Koch et al., 2015). For example, matching networks (Vinyals et al., 2016) can automatically learn discriminative feature representations and a useful distance metric. Therefore, using a 1-NN prediction method, matching networks work well for infrequent labels. However, researchers typically evaluate matching networks on multi-class problems without label imbalance. For EMR coding with extreme label imbalance with several labels occurring thousands of times, traditional parametric neural networks (Kim, 2014) should work very well on the frequent labels. In this chapter, we introduce a new variant of matching networks (Vinyals et al., 2016; Snell et al., 2017a) to address the EMR coding problem. Specifically, we combine the non-parametric idea of kNN and matching networks with traditional neural network text classification methods to handle both frequent and infrequent labels encountered in EMR coding.

Overall, we make the following contributions in this chapter:

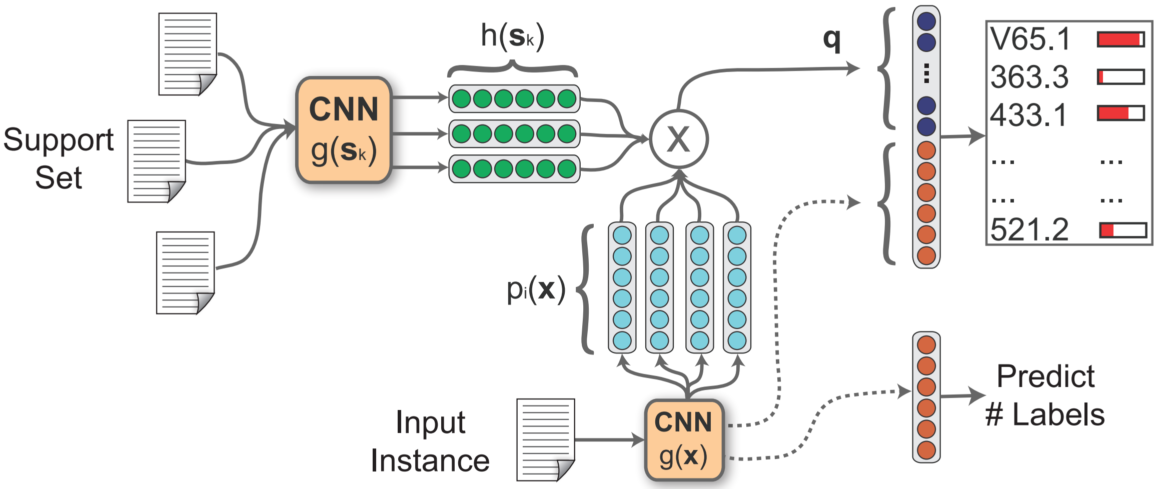
- We propose a novel semi-parametric neural matching network for diagnosis/procedure code prediction from EMR narratives. Our architecture employs ideas from matching networks (Vinyals et al., 2016), multiple attention (Lin et al., 2017), multi-label loss functions (Nam et al., 2014a), and CNNs for text classification (Kim, 2014) to produce a state-of-the-art EMR coding model.
- We evaluate our model on publicly available EMR datasets to ensure reproducibility and benchmarking; we also compare against prior state-of-the-art methods in EMR coding and demonstrate robustness across multiple standard evaluation measures.
- We analyze and measure how each component of our model affects the performance using ablation experiments.

4.1 Related Work: Memory Augmented Neural Networks

Memory networks (Weston et al., 2014) have access to external memory, typically consisting of information the model may use to make predictions. Intuitively, informative memories concerning a given instance are found by the memory network to improve its predictive power. Kamra et al. (2017) use memory networks to fix issues of catastrophic forgetting. They show that external memory can be used to learn new tasks without forgetting previous tasks. Memory networks are now applied to a wide variety of natural language processing tasks, including question answering and language modeling (Sukhbaatar et al., 2015; Bordes et al., 2015; Miller et al., 2016).

Matching networks (Vinyals et al., 2016; Snell et al., 2017a) have recently been developed for few/one-shot learning problems. We can interpret matching networks as a key-value memory network (Miller et al., 2016). The "keys" are training instances, while the "values" are the labels associated with each training example. Intuitively, the concept is similar to a hashmap. The model will search for the most similar training

Figure 4.1: The matching CNN architecture. For each input instance, $\mathbf{x}$, we search a support set using different representations of $\mathbf{x}$ and use the similar support instances and auxiliary features to the output layer.



instance to find its respective "value". Also, matching networks can be interpreted as a kNN based model that automatically learns an informative distance metric. Altae-Tran et al. (2017) used matching networks for drug discovery, a problem where data is limited. Finally, memory networks (Prakash et al., 2017) have recently been used for diagnosis coding. However, we would like to note two significant differences between the memory network from Prakash et al. (2017) and our model. First, they don't use a matching network and their memories rely on extracting information about each label from Wikipedia. In contrast, our model does not use any auxiliary information. Second, they only evaluate on the 50 most frequent labels, while we evaluate on all the labels in the dataset.

4.2 Our Architecture

An overview of our model is shown in Figure 4.1. Our model architecture has two main components.

1. We augment a CNN with external memory over a support set S, which consists of a small subset of the training dataset. The model searches the support set to find similar examples with respect to the input instance. We make use of the homophily assumption that similar instances in the support set are coded with similar labels. Therefore, we use the related support set examples as auxiliary features. The similar instances are chosen automatically by combining ideas

from metric learning and neural attention. We emphasize that unlike in a traditional k-NN setup, we do NOT explicitly use the labels of the support set instances. The support set essentially enriches and complements the features derived from the input instance.

2. Rather than predicting labels by thresholding, we rank them and select the top k labels specific to each instance where k is predicted using an additional output unit (termed MetaLabeler). We train the MetaLabeler along with the classification loss using a multi-task training scheme.

Before we go into more specific details of our architecture, we introduce some notation. Let X represent the set of all training documents and $\mathbf{x}$ be an instance of X. Likewise, let S represent the set of support instances and s be an instance of S. We let L be the total number of unique labels. Our full model is described in following subsections.

4.2.1 Convolutional Neural Networks

Similar to Chapter 3, we use a standard CNN consisting of an embedding layer, a convolution layer, a max-pooling layer, and an output layer (Collobert et al., 2011; Kim, 2014). For consistency with Chapter 3, we represent each instance as

$$g(\mathbf{x}) = \hat{\mathbf{c}}_{\mathcal{W}}$$

where $\hat{\mathbf{c}}_{\mathcal{W}}$ represents the max-pooled feature vectors for the instance $\mathbf{x}$ first defined by Equation 2.7.

4.2.2 Multi-Head Matching Network

Using the support set and the input instance, our goal is to estimate $P(\mathbf{y}|\mathbf{x}, S)$. The support set S is chosen based on nearest neighbors and its selection process is discussed in Section 4.2.4. Among instances in S, our model finds informative support instances with respect to $\mathbf{x}$ and creates a feature vector using them. This feature vector is combined with the input instance to make predictions.

First, each support instance $\mathbf{s}_k \in S$ is projected into the support space using a simple single-layer feed forward NN as

$$h(g(\mathbf{s}_k)) = ReLU(\mathbf{W}_s\, g(\mathbf{s}_k) + \mathbf{b}_s),$$

where $\mathbf{W}_s \in \mathbb{R}^{z \times v}$ and $\mathbf{b}_s \in \mathbb{R}^z$. Likewise, we project each input instance $\mathbf{x}$ into the input space using a different feed forward neural network,

$$p_i(g(\mathbf{x})) = ReLU(\mathbf{W}_\alpha^i \, g(\mathbf{x}) + \mathbf{b}_\alpha^i),$$

where $\mathbf{W}_\alpha^i \in \mathbb{R}^{z \times v}$ and $\mathbf{b}_\alpha^i \in \mathbb{R}^z$. Compared to the support set neural network where we use only a single network, for the input instance we have u projection neural networks. This means we have u versions of $\mathbf{x}$, an idea that is similar to self-attention (Lin et al., 2017), where the model learns multiple representations of an instance. Here each $p_i(g(\mathbf{x}))$ represents a single "head" or representation of the input $\mathbf{x}$. Using different weight matrices, $[\mathbf{W}_\alpha^1, \ldots, \mathbf{W}_\alpha^u]$ and $[\mathbf{b}_\alpha^1, \ldots, \mathbf{b}_\alpha^u]$, we create different representations of $\mathbf{x}$ (multiple heads). For both the input multi-heads and the support instance projection, we note that the same CNN is used (also indicated in Figure 4.1) whose output is subject to the feed forward neural nets outlined thus far in this section.

Rather than searching for a single informative support instance, we search for multiple relevant support instances. For each of the u input instance representations, we calculate a normalized attention score

$$A_{i,k} = \frac{\exp(-d(p_i(g(\mathbf{x})), \, h(g(\mathbf{s}_k)))}{\sum_{\mathbf{s}_{k'} \in S} \left[\exp(-d(p_i(g(\mathbf{x})), \, h(g(\mathbf{s}_{k'})))\right]}$$

where $A_{i,k}$ represents the score of the k-th support example with respect to the i-th input representation $p_i(g(\mathbf{x}))$ and

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2,$$

is the square of the Euclidean distance between the input and support representations.

Next, the normalized scores are aggregated into a matrix $\mathbf{A} \in \mathbb{R}^{u \times |S|}$. Then, we create a feature vector

$$\mathbf{q} = vec(\mathbf{A}\,\mathbf{S}) \tag{4.1}$$

where $\mathbf{q} \in \mathbb{R}^{uz}$, vec is the matrix vectorization operator, and $\mathbf{S} \in \mathbb{R}^{|S| \times z}$ is the support instance CNN feature matrix whose i-th row is $h(g(\mathbf{s}_i))$ for $i = 1, \ldots, |S|$. Intuitively, multiple weighted averages of the support instances are created, one for each of the u input representations. The final feature vector,

$$\mathbf{h} = \mathbf{q} \,||\, g(\mathbf{x}), \tag{4.2}$$

is formed by concatenating the CNN representation of the input instance $\mathbf{x}$ and the support set feature vector $\mathbf{q}$.

Finally, the output layer for L labels involves computing

$$\hat{\mathbf{y}} = P(\mathbf{y}|\mathbf{x}, S) = sigmoid(\mathbf{W}_c\,\mathbf{h} + \mathbf{b}_c) \tag{4.3}$$

where $\mathbf{W}_c \in \mathbb{R}^{L\times(uz+v)}$, $\mathbf{b}_c \in \mathbb{R}^L$. Because we use a sigmoid activation function, each label prediction ($\hat{y}_i$) is in the range from 0 to 1.

4.2.3 MetaLabeler

The easiest method to convert $\hat{\mathbf{y}}$ into label predictions is to simply threshold each element at 0.5. However, most large-scale multi-label problems are highly imbalanced. When training using binary cross-entropy, the threshold 0.5 is optimized for accuracy. Therefore, our predictions will be biased towards 0. A simple way to fix this problem is to optimize the threshold value for each label. Unfortunately, searching for the optimal threshold of each label is computational expensive in large label spaces. Here we train a regression based output layer

$$\hat{r} = ReLU(\mathbf{W}_r\, g(\mathbf{x}) + b_r)$$

where $\hat{r}$ estimates the number of labels $\mathbf{x}$ should be annotated with. At test time, we rank each label by its score in $\hat{\mathbf{y}}$. Next, $\hat{r}$ is rounded to the nearest integer and we predict the top $\hat{r}$ ranked labels.

4.2.4 Training

To train our model, we need to define two loss functions. First, following recent working on multi-label classification with neural networks (Nam et al., 2014b), we train using a multi-label cross-entropy loss. The loss is defined as

$$\mathcal{L}_c = \sum_{i=1}^{L} \big[- y_i \, \log(\hat{y}_i) - (1 - y_i)\, \log(1 - \hat{y}_i)\big],$$

which sums the binary cross-entropy loss for each label. The second loss is used to train the MetaLabeler for which we use the mean squared error

$$\mathcal{L}_{\mathbf{r}} = \|\mathbf{r} - \hat{\mathbf{r}}\|_2^2$$

where $\mathbf{r}$ is the vector of correct numbers of labels and $\hat{\mathbf{r}}$ is our estimate. We train these two losses using a multi-task learning paradigm (Collobert et al., 2011).

Similar to previous work with matching networks (Vinyals et al., 2016; Snell et al., 2017a), "episode" or mini-batch construction can have an impact on performance. In the multi-label setting, episode construction is non-trivial. We propose a simple strategy for choosing the support set S which we find works well in practice. First, at the beginning of the training process we loop over all training examples and store $g(\mathbf{x})$ for every training instance. We will refer to this set of vectors as T. Next, for every step of the training process (for every mini-batch M), we search $T \setminus M$ to find the e nearest neighbors (using Euclidean distance) per instance to form our support set S. Likewise, we add e random examples from $T \setminus M$ to the support set. Therefore, our support set S contains up to $|M|e + e$ instances. The purpose of the random examples is to ensure the distance metric learned during training (captured by improving representations of documents as influenced by all network parameters) is robust to noisy examples.

4.2.5 Matching Network Interpretation

If we do not use the support set label vectors, then what is our network learning? To answer this question we directly compare the matching network formulation to our method. Matching networks can be expressed as

$$\hat{\mathbf{y}} = \sum_{\mathbf{s}_k \in S} a(\mathbf{x}, \mathbf{s}_k)\, \mathbf{y}_{\mathbf{s}_k}$$

where $a(,)$ is the attention/distance learned between two instances, k indexes each support instance, and $\mathbf{y}_k$ is a one-hot encoded vector. $a(,)$ is equivalent to $A_{1,k}$ assuming we use a single head. Traditional matching networks use one-hot encoded vectors because they are evaluated on multi-class problems. EMR coding is a multi-label problem. Hence, $\mathbf{y}_k$ is a multi-hot encoded vector. Moreover, with thousands of labels, it is unlikely even for neighboring instance pairs to share many labels; this problem is not encountered in the multi-class setting. We overcome this issue by learning new output label vectors for each support set instance. Assuming a single head, our method can be re-written as

$$\hat{\mathbf{y}} = sigmoid(\mathbf{W}_c^1\, g(\mathbf{x}) + \mathbf{b}_c + \sum_{\mathbf{s}_k \in S} a(\mathbf{x}, \mathbf{s}_k)\, \tilde{\mathbf{y}}_{\mathbf{s}_k}), \tag{4.4}$$

where $\tilde{\mathbf{y}_k}$ is the learned label vector for support instance s. Next, we define $\tilde{\mathbf{y}_k}$, the

Table 4.1: This table presents the number of training examples (# Train), the number of test examples (# Test), label cardinality (LC), and the average number of instances per label (AI/L) for the MIMIC II and MIMIC III datasets.

	# Train	# Test	# Labels	LC	AI/L
MIMIC II	18822	2282	7042	36.7	118.8
MIMIC III	37016	2755	6932	13.6	80.8

learned support set vectors, as

$$\tilde{\mathbf{y}}_{\mathbf{s}_k} = \mathbf{W}_c^2\, h(g(\mathbf{s}_k)) \tag{4.5}$$

where both $\mathbf{W}_c^1$ and $\mathbf{W}_c^2$ are submatrices of $\mathbf{W}_c$. Using this reformulation, we can now see that our method's main components (equations (4.1)-(4.3)) are equivalent to this more explicit matching network formulation (equations (4.4)–(4.5)). Intuitively, our method combines a traditional output layer – the first half of equation 4.4 – with a matching network where the support set label vectors are learned to better match the labels of their nearest neighbors.

4.3 Experiments

In this section we compare our work with prior state-of-the-art medical coding methods. In Section 4.3.1 we discuss the two publicly available datasets we use and describes the implementation details of our model. We summarize the various baselines and models we compare against in Section 4.3.3. The evaluation metrics are described in Section 4.3.4. Finally, we discuss how our method performs in Section 4.3.5.

4.3.1 Datasets

EMR data is generally not available for public use especially if it involves textual notes. Therefore, we focus on the publicly available Medical Information Mart for Intensive Care (MIMIC) datasets for benchmarking purposes. We evaluate using two versions of MIMIC: MIMIC II (Lee et al., 2011) and MIMIC III (Johnson et al., 2016), where the former is a relatively smaller and older dataset and the latter is the most recent version. Following prior work (Perotte et al., 2013; Vani et al., 2017), we use the free text discharge summaries in MIMIC to predict the ICD-9-CM codes. The dataset statistics are shown in Table 4.1.

For comparison purposes, we use the same MIMIC II train/test splits as Perotte et al. (2013). Specifically, we use discharge reports collected from 2001 to 2008 from the intensive care unit (ICU) of the Beth Israel Deaconess Medical Center and predict both diagnosis and procedure codes. Following Perotte et al. (2013), the labels for each discharge summary are extended using the parent of each label in label set. The parents are based on the ICD-9-CM hierarchy. We use the hierarchical label expansion to maximize the prior work we can compare against.

The MIMIC III dataset has been extended to include health records of patients admitted to the Beth Israel Deaconess Medical Center from 2001 to 2012 and hence provides a test bed for more advanced learning methods. Unfortunately, it does not have a standard train/test split to compare against prior work given we believe we are the first to look at it for this purpose. Hence, we use both MIMIC II and MIMIC III for comparison purposes. Furthermore, we do not use the hierarchical label expansion on the MIMIC III dataset and we only predict diagnosis codes for MIMIC III.

Before we present our results, we discuss an essential distinction between the MIMIC II and MIMIC III datasets. Particularly, we are interested in the differences concerning label imbalance. From Table 4.1, we find that MIMIC III has almost twice as many examples compared to MIMIC II in the dataset. However, MIMIC II on average has more instances per label. Thus, although MIMIC III has more examples, each label is used fewer times on average compared to MIMIC II. The reason for this is because of how the label sets for each instance were extended using the ICD-9 hierarchy in MIMIC II.

4.3.2 Implementation Details

Preprocessing: Each discharge summary was tokenized using a simple regex tokenization scheme (\w\w+). Also, each word/token that occurs less than five times in the training dataset was replaced with the *UNK* token.

Model Details: For our CNN, we used convolution filters of size 3, 4 and 5 with 300 filters for each filter size. We used 300 dimensional skip-gram (Mikolov et al., 2013b) word embeddings pre-trained on PubMed. The Adam optimizer (Kingma and Ba, 2014) was used for training with the learning rate 0.0001. The mini-batch size was set to 4, e – the number of nearest neighbors per instance – was set to 16, and the number of heads (u) is set to 8. Our code is available at: `https://github.com/bionlproc/med-match-cnn`

4.3.3 Baseline Methods

In this chapter, we focused on comparing our method to state-of-the-art methods for diagnosis code prediction such as grounded recurrent neural networks (Vani et al., 2017) (GRNN) and multi-label CNNs (Baumel et al., 2017). We also compare against traditional binary relevance methods where independent binary classifiers (L1-regularized linear models) are trained for each label. Next, we compare against hierarchical SVM (Perotte et al., 2013), which incorporates the ICD-9-CM label hierarchy. Finally, we also report the results of the traditional matching network with one modification: We train the matching network with the multi-label loss presented in Section 4.2.4 and threshold using the MetaLabeler described in Section 4.2.3.

We also present two versions of our model: *Match-CNN* and *Match-CNN Ens*. Match-CNN is the multi-head matching network introduced in Section 4.2. Match-CNN Ens is an ensemble that averages three Match-CNN models, each initialized using a different random seed.

4.3.4 Evaluation Metrics

We evaluate our method using a wide variety of standard multi-label evaluation metrics. We use the popular micro and macro averaged F1 measures to assess how our model (with the MetaLabeler) performs when thresholding predictions. For problems with large labels spaces that suffer from significant imbalances in label distributions, the default threshold of 0.5 generally performs poorly (hence our use of MetaLabeler). To remove the thresholding effect bias, we also report different versions of the area under the precision-recall (AUPRC) and receiver operating characteristic (AUCOC) curves. Finally, in a real-world setting, our system would not be expected to replace medical coders. We would expect medical coders to use our system to become more efficient in coding EMRs. Therefore, we would rank the labels based on model confidence and medical coders would choose the correct labels from the top few. To understand if our system would be useful in a real-world setting, we evaluate with precision at k (P@k) and recall at k (R@k). Having high P@k and R@k are critical to effectively encourage the human coders to use and benefit from the system. These evaluation metrics are explained in Chapter 2.

4.3.5 Results

We show experimental results on MIMIC II in Table 4.2. Overall, our method improves on prior work across a variety of metrics. With respect to micro F1, we

Table 4.2: Results for the MIMIC II dataset. Models marked with * represent our custom implementations.

			F1		AUC (PR)		AUC (ROC)		P@k		R@k	
	Prec.	Recall	Micro	Macro	Micro	Macro	Micro	Macro	8	40	8	40
Flat SVM (Perotte et al., 2013)	**86.7**	16.4	27.6	–	–	–	–	–	–	–	–	–
Hier. SVM (Perotte et al., 2013)	57.7	30.1	39.5	–	–	–	–	–	–	–	–	–
Logistic (Vani et al., 2017)	77.4	39.5	52.3	04.2	54.1	12.5	91.9	70.4	91.3	57.2	16.9	52.8
Attn BoW (Vani et al., 2017)	74.5	39.9	52.0	02.7	52.1	07.9	97.5	80.7	91.2	54.9	16.9	50.8
GRU-128 (Vani et al., 2017)	72.5	39.6	51.2	02.7	52.3	08.2	97.6	**82.7**	90.6	54.1	16.8	50.1
BiGRU-64 (Vani et al., 2017)	71.5	36.7	48.5	02.1	49.3	07.1	97.3	81.1	89.2	52.2	16.5	48.3
GRNN-128 (Vani et al., 2017)	75.3	47.2	58.0	05.2	58.7	12.6	97.6	81.5	93.0	59.2	17.2	54.8
BiGRNN-64 (Vani et al., 2017)	76.1	46.6	57.8	05.4	58.9	13.1	97.5	79.8	92.5	**59.6**	17.2	55.2
CNN (Baumel et al., 2017) *	81.0	40.3	53.8	03.1	59.9	12.7	97.1	75.9	93.1	58.5	20.7	58.6
Matching Network *	43.9	38.8	41.2	01.4	39.4	03.4	89.3	55.1	79.3	42.7	17.2	42.5
Match-CNN (Ours)	60.5	56.1	58.2	06.4	61.2	14.8	**97.7**	79.2	93.0	58.6	20.7	59.0
Match-CNN Ens. (Ours)	61.6	**56.7**	**59.1**	**06.6**	**62.3**	**15.7**	**97.7**	79.3	**93.5**	59.4	**20.8**	**59.8**

Table 4.3: Results for the MIMIC III dataset. Models marked with * represent our custom implementations.

			F1		AUC (PR)		AUC (ROC)		P@k		R@k	
	P	R	Micro	Macro	Micro	Macro	Micro	Macro	8	40	8	40
Logistic (Vani et al., 2017) *	71.1	24.2	36.1	02.6	41.9	**14.7**	96.1	75.1	55.4	**21.1**	41.4	**68.6**
CNN (Baumel et al., 2017) *	**72.6**	24.6	36.7	02.1	37.6	09.5	94.2	69.7	53.4	19.6	39.5	63.6
Matching Network *	24.8	23.7	24.2	00.8	18.3	02.8	82.3	55.4	31.0	12.8	23.1	43.1
Match-CNN (Ours)	46.6	44.7	45.6	04.1	42.1	11.9	96.3	72.6	55.7	20.6	41.3	67.0
Match-CNN Ens. (Ours)	48.8	**44.9**	**46.8**	**04.3**	**44.1**	12.9	**96.5**	**76.0**	**57.0**	**21.1**	**42.2**	68.3

improve upon GRNN-128 by over 1%. Also, while macro-F1 is still low in general, we also improve macro F1 compared to state-of-the-art neural methods by more than 1%. In general, both micro and macro F1 are highly dependent on the thresholding methodology. Rather than thresholding at 0.5, we rank the labels and pick the top *k* based on a trained regression output layer. Can we do better than using a MetaLabeler? To measure this, we look at the areas under PR/ROC curves. Regarding micro and macro AUPRC, we improve on prior work by $\approx 2.5\%$. This suggests that via better thresholding, the chances of improving both micro and macro F1 are higher for Match-CNN compared to other methods. Finally, we are also interested in metrics that evaluate how this model would be used in practice. We perform comparably with prior work on P@k. We show strong improvements in R@k with over a 4% improvement in R@40 compared to grounded RNNs and over 1% improvement when compared with Baumel et al. (2017). Our method also outperforms matching networks across every evaluation measure.

We present MIMIC III results in Table 4.3. We reiterate that MIMIC III does not have a standard train/test split. Hence we compare our model to our implementations

Table 4.4: Ablation results for the MIMIC III dataset.

	F1		P@k		R@k		AUC (PR)	
	Micro	Macro	8	40	8	40	Micro	Macro
Match-CNN	45.6	04.1	55.7	20.6	41.3	67.0	42.1	11.9
- Matching	42.9	03.4	53.4	19.6	39.5	63.6	37.6	09.5
- MetaLabler	39.1	02.6	55.7	20.6	41.3	67.0	42.1	11.9
- Multi-Head	45.0	03.4	54.8	20.2	40.3	65.6	41.7	11.3

of methods from prior efforts. For MIMIC III also we show improvements in multiple evaluation metrics. Interestingly, our method performs much better than the standard CNN on MIMIC III, compared to the relative performances of the two methods on MIMIC II. Match-CNN improves on CNN in R@40 by almost 5% on the MIMIC III dataset. The gain in R@40 is more than the 1% improvement found on MIMIC II. We hypothesize that the improvements on MIMIC III are because the label imbalance found in MIMIC III is higher than MIMIC II. Increased label imbalances mean more labels occur less often. Therefore, we believe our model works better with less training examples per label compared to the standard CNN model.

In Table 4.4 we analyze each component of our model using an ablation analysis on the MIMIC III dataset. First, we find that removing the matching component significantly effects our performance by reducing micro AUPRC by almost 5%. Regarding micro and macro F1, we also notice that the MetaLabeler heuristic substantially improves on default thresholding (0.5). Finally, we see that the multi-head matching component provides reasonable improvements to our model across multiple evaluation measures. For example, P@8 and P@40 decrease by around 1% when we use attention with a single input representation.

4.4 Conclusion

In this chapter, we introduced a semi-parametric multi-head matching network and applied it to EMR coding datasets. We find that by combining the non-parametric properties of matching networks with a traditional classification output layer, we improve metrics for both frequent and infrequent labels in the dataset. However, this model does not handle ICD-9 codes that never appeared in the dataset. Furthermore, with the evaluation strategy used in this chapter, it is ambiguous how the model performs on infrequent codes compared to codes that occur infrequently. We address theses issues in Chapter 5.

Chapter 5 Zero-shot and Few-shot Multi-label Learning

As discussed in Chapter 1, there are two major difficulties when developing machine learning methods for large-scale multi-label text classification problems. First, the documents may be long, sometimes containing more than a thousand words (Mullenbach et al., 2018). Finding the relevant information in a large document for a specific label results in needle in a haystack situation. Second, data sparsity is a common problem; as the total number of labels grows, a few labels may occur frequently, but most labels will occur infrequently. Rubin et al. (2012) refer to datasets that have long-tail frequency distributions as "power-law datasets". Methods that predict infrequent labels fall under the paradigm of few-shot classification which refers to supervised methods in which only a few examples, typically between 1 and 5, are available in the training dataset for each label. With predefined label spaces, some labels may never appear in the training dataset. Zero-shot problems extend the idea of few-shot classification by assuming no training data is available for the labels we wish to predict at test time. In this chapter, we explore both of these issues, long documents and power-law datasets, with an emphasis on analyzing the few- and zero-shot aspects of large-scale multi-label problems.

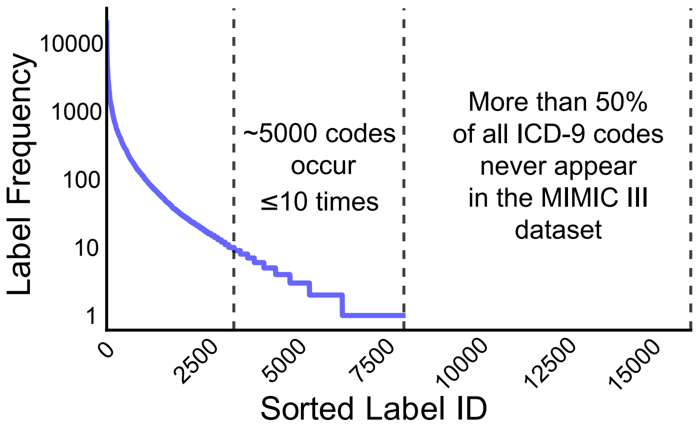
In Figure 5.1, we plot the label frequency distribution of diagnosis and procedure labels for the entire MIMIC III (Johnson et al., 2016) dataset. A few labels occur more than 10,000 times, around 5000 labels occur between 1 and 10 times, and of the 17,000 diagnosis and procedure labels, more than 50% never occur. There are a few reasons a label may never occur in the training dataset. In healthcare, several disorders are rare; therefore corresponding labels may not have been observed yet in a particular clinic. Sometimes new labels may be introduced as the field evolves leading to an *emerging label* problem. This is intuitive for applications such as hashtag prediction on Twitter. For example, last year it would not have made sense to annotate tweets with the hashtag #EMNLP2018. Yet, as this year's conference approaches, labeling tweets with the #EMNLP2018 will help users find relevant information.

How does the emerging code problem appear in health care? Do many new codes get added to standardized medical coding terminologies? The World Health Organization (WHO) recently introduced the "gaming disorder" as a mental health disorder in ICD-11[1]. It was estimated that nearly 1 in 10 adolescents aged 8 to 18 suffer from

---

[1] `http://www.who.int/features/qa/gaming-disorder/en/`

Figure 5.1: This plot shows the label frequency distribution of ICD-9 codes in MIMIC III.



gaming addiction (Gentile, 2009). With ICD-10, patients with gaming addictions may be annotated with the label "Impulse-control disorder". While some scholars suggest that the estimates of video game addiction may be inflated (Wood, 2008), in the event that hospitals switch to ICD-11, it is possible that many medical records will begin to be annotated with the "gaming disorder" term. Switching to a new vocabulary does not happen overnight. The word health organization released the full ICD-10 terminology in 1994. The US Department of Health and Human Services did not mandate the health industry to transition from ICD-9 to ICD-10 until 2015. However, when we transition to a new vocabulary, and given the introduction of new soon-to-be frequent codes, it is important to develop methods that can predict these codes until enough training data is collected to take advantage of traditional supervised methods. This is evident given how many new codes were added to ICD-10 compared to ICD-9. Specifically, ICD-9 contains just over 14 thousand diagnosis codes compared to the 68 thousand codes in ICD-10. Furthermore, the number of procedure codes increased nearly 19 times from almost 4 thousand codes in ICD-9 to over 70 thousand in ICD-10.

Infrequent labels may not contribute heavily to the overall accuracy of a multi-label model but in some cases correct prediction of such labels is crucial but not straightforward. For example, in assigning diagnosis labels to EMRs, it is important that trained human coders are both accurate and thorough. Errors may cause unfair financial burden on the patient. Coders may have an easier time assigning frequent labels to EMRs because they are encountered more often. Also, frequent labels are generally easier to predict using machine-learning based methods. However, infrequent or obscure labels will be easily confused or missed causing billing mistakes and/or causing the coders to spend more time annotating each record. Thus, we believe methods that handle infrequent and unseen labels in the multi-label setting are important.

Current evaluation methods for large-scale multi-label classification mostly ignore infrequent and unseen labels. Popular evaluation measures focus on metrics such as micro-F1, recall at k (R@k), precision at k (P@k), and macro-F1. As it is well-known that micro-F1 gives more weight to frequent labels, papers on this topic also report macro-F1, the average of label-wise F1 scores, which equally weights all labels. Unfortunately, macro-F1 scores are generally low and the corresponding performance differences between methods are small. Moreover, it is possible to improve macro-F1 by only improving a model's performance on frequent labels, further confounding its interpretation. Hence we posit that macro-F1 is not enough to compare large-scale multi-label learning methods on infrequent labels and it does not directly evaluate zero-shot labels. Here, we take a step back and ask: can the model predict the correct few-shot (zero-shot) labels from the set of all few-shot (zero-shot) labels? To address this, we test our approach by adapting the *generalized zero-shot classification* evaluation methodology by Xian et al. (2017) to the multi-label setting.

In this chapter, we propose and evaluate a neural architecture suitable for handling few- and zero-shot labels in the multi-label setting where the output label space satisfies two constraints: (1). the labels are connected forming a DAG and (2). each label has a brief natural language descriptor. These assumptions hold in several multi-label scenarios including assigning diagnoses/procedures to EMRs and indexing biomedical articles with medical subject headings. Taking advantage of this prior knowledge on labels is vital for zero-shot prediction. Specifically, using the EMR coding use-case, we make the following contributions:

1. We overcome issues arising from processing long documents by introducing a new neural architecture that expands on recent attention-based CNNs (ACNN (Mullenbach et al., 2018)). Our model learns to predict few- and zero-shot labels by matching discharge summaries in EMRs to feature vectors for each label obtained by exploiting structured label spaces with graph CNNs (GCNN (Kipf and Welling, 2017)).

2. We provide a fine-grained evaluation of state-of-the-art EMR coding methods for frequent, few-shot, and zero-shot labels. By evaluating power-law datasets using an extended generalized zero-shot methodology that also includes few-

shot labels, we present a nuanced analysis of model performance on infrequent ICD-9-CM codes.

5.1 Related Work

In this section, we present related-work about three areas of research relevant to this chapter: Few- and zero-shot learning, structured label correlations for multi-label classification, and GCNNs.

5.1.1 Few-Shot and Zero-Shot Learning

While neural networks are generally considered to need large datasets, they have been shown to work well on few-shot classification tasks. To handle infrequent labels, most NN methods use a k-NN-like approach. Siamese NNs (Koch et al., 2015) learn a nonlinear distance metric using a pairwise loss function. Matching networks (Vinyals et al., 2016) introduce an instance-level attention method to find relevant neighbors. Prototypical Networks (Snell et al., 2017b) average all instances in each class to form "prototype label vectors" and train using a traditional cross-entropy loss.

Zero-shot learning has not been widely explored in the large-scale multi-label classification scenario. Like neural few-shot methods, neural zero-shot methods use a matching framework. Instead of matching input instances with other instances, they are matched to predefined label vectors. For example, the Attributes and Animals Dataset (Xian et al., 2017) contains images of animals and the label vectors consist of features describing the types of animals (e.g., stripes: yes). When feature vectors for labels are not available, the average of the pretrained word embeddings of the class names have been used. The attribute label embedding method (Akata et al., 2016) uses a pairwise ranking loss to match zero-shot label vectors to instances. (Romera-Paredes and Torr, 2015) introduced the "embarrassingly simple zero-shot learning" (ESZSL) method which is trained using a mean squared error loss. A few zero-shot methods do not translate well to multi-label problems. CONSE (Mikolov et al., 2013a) averages the embeddings for the top predicted supervised label vectors to match to zero-shot label vectors. CONSE assumes that both supervised and zero-shot labels cannot be assigned to the same instance. In this chapter, we expand on the generalized zero-shot evaluation methodology introduced by (Xian et al., 2017) to large-scale multi-label classification. Finally, it is important to note that zero-shot classification has been previously studied in the multi-label setting (Mensink et al., 2014). However, they focus on image classification and their datasets only contain around 300 labels.

5.1.2 Structured Label Correlations for Multi-label Classification

Ontological resources that consist of hyperemic ("is a") and meronymic ("part of") relations between labels are common in many domains including e-commerce and biomedicine. Such hierarchical knowledge sources, when available, are commonly exploited to improve multi-label predictions. Hierarchical binary relevance (HBR) constructs a dataset for each node in the hierarchy consisting of instances belonging to its parent (Tsoumakas et al., 2010). Predictions can be made using a top-down approach where an instance won't proceed to the inner nodes of a subtree if its root node does not positively classify the instance. This suggests that for large label spaces it is not necessary to run every classifier for each label. There are well known weaknesses to such an approach the most significant one being the so called "blocking problem" where recall is adversely affected by higher nodes ruling out correct downstream predictions.

Recent work has transformed the hierarchical classification problem into an optimal subgraph search problem (Bi and Kwok, 2011). In the BR framework, the key idea is to independently pick the top k labels based on each classifier score. Instead, the optimal subgraph respects specific properties that ensure the consistency of the directed asyclic graph (DAG). This is different from HBR which builds a new dataset for each node. Label co-occurrence as well as semantic label hierarchy have been formulated via label ranking optimization problems (Wu et al., 2015) by encouraging parent labels to rank higher than their children.

5.1.3 Graph Convolutional Neural Networks

GCNNs generalize CNNs beyond 2d and 1d spaces. Defferrard et al. (2016) developed spectral methods to perform efficient graph convolutions. Kipf and Welling (2017) assume a graph structure is known over input instances and apply GCNNs for semi-supervised learning. GCNNs are applied to relational data (e.g., link prediction) by Schlichtkrull et al. (2018). GCNNs have also had success in many NLP applications including, but not limited to, semantic role labeling (Marcheggiani and Titov, 2017), dependency parsing (Strubell and McCallum, 2017), and machine translation (Bastings et al., 2017).

There are three GCNN papers that share similarities with our work.

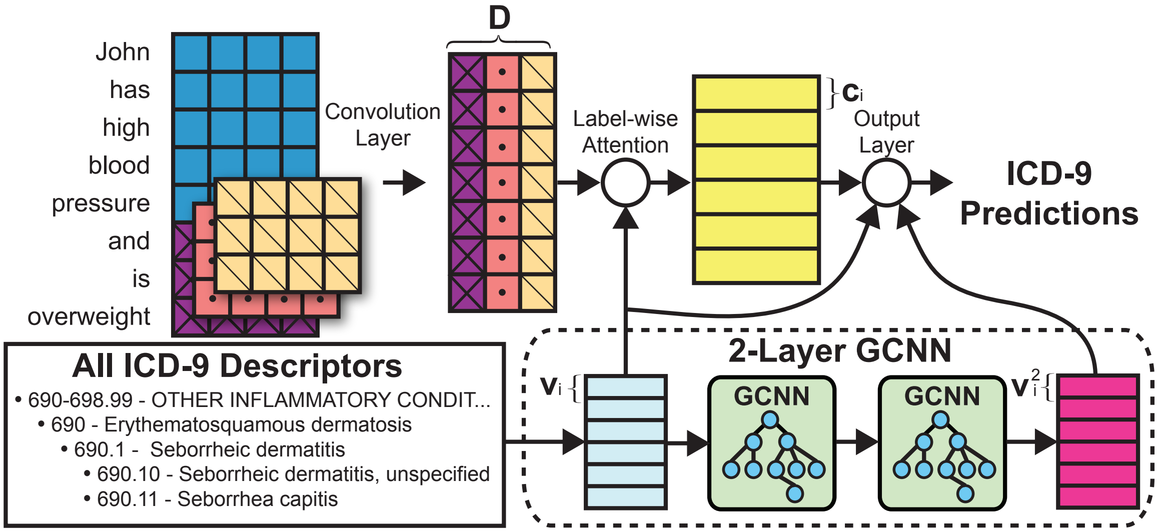

D
John
has
high
blood
pressure
and
is
overweight
Convolution Layer
Label-wise Attention
c_i
Output Layer
ICD-9 Predictions
All ICD-9 Descriptors
• 690-698.99 - OTHER INFLAMMATORY CONDIT...
• 690 - Erythematosquamous dermatosis
• 690.1 - Seborrheic dermatitis
• 690.10 - Seborrheic dermatitis, unspecified
• 690.11 - Seborrhea capitis
2-Layer GCNN
v_i
GCNN
GCNN
v_i^2

Figure 5.2: Overview of our architecture

1. Peng et al. (2018) use a GCNN on a word co-occurrence graph for large-scale text classification where the GCNN operates on documents/words, while our GCNN operates on the labels.

2. Chen et al. (2017) use GCNNs on structured label spaces. However, their experiments focus on smaller label spaces and do not handle/assess zero-shot and few-shot labels. Also, their experiments for text classification do not incorporate attention and simply use an average of word vectors to represent each document.

3. Wang et al. (2018) propose a zero-shot GCNN image classification method for structured multi-class problems. We believe their method may transfer to the multi-label text classification setting but exact modifications to affect that are not clear (i.e., their semi-supervised approach may not be directly applicable). Likewise, porting to text is nontrivial for long documents.

5.2 Method

Figure 5.2 shows the overall schematic of our architecture. Intuitively, we incorporate four main components. First, we assume we have the full English descriptor/gloss for each label we want to predict. We form a vector representation for each label by averaging the word embeddings for each word in its descriptor. Second, the label vectors formed from the descriptor are used as attention vectors (label-wise attention) to find the most informative ngrams in the document for each label. For each label, this will produce a separate vector representation of the input document. Third, the label vectors are passed through a two layer GCNN to incorporate hierarchical information about the label space. Finally, the vectors returned from the GCNN are matched to the document vectors to generate predictions.

5.2.1 Convolutional Neural Network

Contrary to Chapters 3 and 4, instead of using a max-over-time pooling layer, we learn to find relevant ngrams in a document for each label via label-wise attention (Mullenbach et al., 2018). The CNN will return a document feature matrix $\mathbf{D} \in \mathbb{R}^{(n-s+1)\times u}$ where each column of $\mathbf{D}$ is a feature map, u is the total number of convolution filters, n is the number of words in the document, and s is the width of convolution filters.

5.2.2 Label Vectors

To be able to predict labels that were not in the training dataset, we avoid learning label specific parameters. We use the label descriptors to generate a feature vector for each label. First, to preprocess each descriptor, we lowercase all words and remove stop-words. Next, each label vector is formed by averaging the remaining words in the descriptor

$$\mathbf{v}_i = \frac{1}{|N|}\sum_{j\in N}\mathbf{w}_j, \qquad i = 1,\ldots,L, \tag{5.1}$$

where $\mathbf{v}_i \in \mathbb{R}^d$, L is the number of labels, and N is the index set of the words in the descriptor. Prior zero-shot work has focused on projecting input instances into the same semantic space as the label vectors (Sandouk and Chen, 2016). For zero-shot image classification, this is a non-trivial task. Because we work with textual data, we simply share the word embeddings between the convolutional layer and the label vector creation step to form $\mathbf{v}_i$.

5.2.3 Label-Wise Attention

Similar to the work by Mullenbach et al. (2018), we employ label-wise attention to avoid the needle in the haystack situation encountered with long documents. But here we also need to find relevant information for zero-shot classes. So we use the label vectors $\mathbf{v}_i$ rather than learning label specific attention parameters. First, we pass the document feature matrix $\mathbf{D}$ through a simple feed-forward neural network

$$\mathbf{D}^2 = \tanh(\mathbf{D}\,\mathbf{W}_b + \mathbf{b}_b)$$

where $\mathbf{W}_b \in \mathbb{R}^{u \times d}$ and $\mathbf{b}_b \in \mathbb{R}^d$. This mapping is important because the dimensionality of the ngram vectors (rows) in $\mathbf{D}$ depends on u, the number of scores we generate for each ngram. Given $\mathbf{D}^2$, we generate the label-wise attention vector

$$\mathbf{a}_i = softmax(\mathbf{D}^2 \, \mathbf{v}_i), \quad i = 1, \ldots, L, \tag{5.2}$$

where $\mathbf{a}_i \in \mathbb{R}^{n-s+1}$ measures how informative each ngram is for the i-th label. Finally, we use $\mathbf{D}$, and generate L label-specific document vector representations

$$\mathbf{c}_i = \mathbf{a}_i^T \, \mathbf{D}, \quad i = 1, \ldots, L,$$

such that $\mathbf{c}_i \in \mathbb{R}^u$. Intuitively, $\mathbf{c}_i$ is the weighted average of the rows in $\mathbf{D}$ forming a vector representation of the document for the i-th label.

5.2.4 GCNN Output Layer

Traditionally, the output layer of a CNN would learn label specific parameters optimized via a cross-entropy loss. Instead, our method attempts to match documents to their corresponding label vectors. In essence, this becomes a retrieval problem. Before using each document representation $\mathbf{c}_i$ to score its corresponding label, we take advantage of the structured knowledge we have over our label space using a 2-layer GCNN. For both the MIMIC II and MIMIC III datasets, this information is hierarchical. A snippet of the hierarchy can be found in Figure 5.2.

Starting with the label vectors $\mathbf{v}_i$, we combine the label vectors of the children and parents for the i-th label to form

$$\mathbf{v}_i^1 = f(\mathbf{W}^1 \mathbf{v}_i + \sum_{j \in \mathcal{N}_p} \frac{\mathbf{W}_p^1 \mathbf{v}_j}{|\mathcal{N}_p|} + \sum_{j \in \mathcal{N}_c} \frac{\mathbf{W}_c^1 \mathbf{v}_j}{|\mathcal{N}_c|} + \mathbf{b}_g^1)$$

where $\mathbf{W}^1 \in \mathbb{R}^{q \times d}$, $\mathbf{W}_p^1 \in \mathbb{R}^{q \times d}$, $\mathbf{W}_c^1 \in \mathbb{R}^{q \times d}$, $\mathbf{b}_g^1 \in \mathbb{R}^q$, f is the rectified linear unit (Nair and Hinton, 2010) function, and $\mathcal{N}_c$ ($\mathcal{N}_p$) is the index set of the i-th label's children (parents). We use different parameters to distinguish each edge type. In this chapter, given we only deal with hierarchies, the edge types include edges from parents, from children, and self edges. This can be adapted to arbitrary DAGs, where parent edges represent all incoming edges and the child edges represent all outgoing edges for each node.

The second layer follows the same formulation as the first layer with

$$\mathbf{v}_i^2 = f(\mathbf{W}^2\mathbf{v}_i^1 + \sum_{j\in\mathcal{N}_p} \frac{\mathbf{W}_p^2\mathbf{v}_j^1}{|\mathcal{N}_p|} + \sum_{j\in\mathcal{N}_c} \frac{\mathbf{W}_c^2\mathbf{v}_j^1}{|\mathcal{N}_c|} + \mathbf{b}_g^2)$$

where $\mathbf{W}^2 \in \mathbb{R}^{q\times q}$, $\mathbf{W}_p^2 \in \mathbb{R}^{q\times q}$, $\mathbf{W}_c^2 \in \mathbb{R}^{q\times q}$, and $\mathbf{b}_g^2 \in \mathbb{R}^q$. Next, we concatenate both the averaged description vector (from equation (5.1)) with the GCNN label vector

$$\mathbf{v}_i^3 = \mathbf{v}_i \mid\mid \mathbf{v}_i^2,$$

where $\mathbf{v}_i^3 \in \mathbb{R}^{d+q}$. Now, to compare the final label vector $\mathbf{v}_i^3$ with its document vector $\mathbf{c}_i$, we transform the document vector into

$$\mathbf{e}_i = ReLU(\mathbf{W}_o\mathbf{c}_i + \mathbf{b}_o), \quad i = 1, \ldots, L,$$

where $\mathbf{W}_o \in \mathbb{R}^{(q+d)\times u}$ and $\mathbf{b}_o \in \mathbb{R}^{q+d}$. This transformation is required to match the dimension to that of $\mathbf{v}_i^3$. Finally, the prediction for each label i is generated via

$$\hat{y}_i = sigmoid(\mathbf{e}_i^T\,\mathbf{v}_i^3), \quad i = 1, \ldots, L.$$

During experiments, we found that using either the output layer GCNN or a separate GCNN for the attention vectors (equation (5.2)) did not result in an improvement and severely slowed convergence.

5.2.5 Training

Following the same strategy as Chapters 3 and 4, we train our model using a multi-label binary cross-entropy loss (Nam et al., 2014a)

$$\mathcal{L} = \sum_{i=1}^{L} \big[- y_i \, \log(\hat{y}_i) - (1 - y_i) \, \log(1 - \hat{y}_i)\big],$$

where $y_i \in \{0, 1\}$ is the ground truth for the i-th label and $\hat{y}_i$ is our sigmoid score for the i-th label.

5.3 Experiments

In this chapter, we use the same two medical datasets for evaluation purposes from Chapter 4: MIMIC II (Jouhet et al., 2012) and MIMIC III (Johnson et al., 2016).

Table 5.1: MIMIC II and MIMIC III dataset statistics for few- and zero-shot learning.

	Number of Labels		
Dataset	Frequent (S)	Few-Shot (F)	Zero-Shot (Z)
MIMIC II	3228	3459	355
MIMIC III	4403	4349	178

Both datasets contain discharge summaries annotated with a set of ICD-9 diagnosis and procedure labels. Furthermore, we use the same train/test splits as Chapter 4. Discharge summaries are textual documents consisting of, but not limited to, physician descriptions of procedures performed, diagnoses made, the patient's medical history, and discharge instructions. We use the label descriptors provided by the world health organization (WHO) to generate label vectors, whose average descriptor length is seven words. Following a generalized zero-shot learning evaluation methodology (Xian et al., 2017), we split the ICD-9 labels into three groups based on frequencies in the training dataset: The frequent group S that contains all labels that occur > 5 times, the few-shot group F that contains labels that occur between 1 and 5 times, and the zero-shot group Z of labels that never occurred. The groups are only used for evaluation. That is, during training, systems are *optimized over all labels simultaneously*. Also, instances that do not contain few- or zero-shot classes are removed from their respective groups during evaluation. This grouping is important to assess how each model performs across labels grouped by label frequency. Our evaluation methodology differs from that of (Xian et al., 2017) in two ways. First, because each instance is labeled with multiple labels, the same instance can appear in all groups — S, F, and Z. Second, instead of top-1 accuracy or HIT@k evaluation measures, we focus on R@k to handle multiple labels. At a high level, we want to examine whether a model can distinguish the correct few-shot (zero-shot) labels from the set of all few-shot (zero-shot) labels. Therefore, the R@k measures in Tables 5.2 and 5.3, and Figure 5.3 are computed relative to each group.

5.3.1 Datasets

We use the same datasets as Chapter 4. The number of labels in the frequent, few- and zero-shot groups in the MIMIC II and III datasets are reported in Table 5.1. For reproducibility purposes, we use the same training/test splits of the MIMIC II as Perotte et al. (2013). Following the procedures in Perotte et al. (2013), Vani et al. (2017), and Chapter 4, for each diagnosis and procedure label assigned to each medical report, we add its parents using the ICD-9 hierarchy. Each report in MIMIC II is annotated with nearly 37 labels on average using hierarchical label expansion.

MIMIC III does not contain a standardized training/test split. Therefore, we create our own split that ensures the same patient does not appear in both the training and test datasets. Unlike the MIMIC II dataset, we do not augment the labels using the ICD-9 hierarchy. Also, contrary to Chapter 4, we predict both diagnosis and procedure codes. The ICD-9 hierarchy has three main levels. For MIMIC III, level 0 labels make up about 5% of all occurrences, level 1 labels make up about 62%, and level 2 (leaf level) labels make up about 33%. Also, each MIMIC III instance contains16 ICD-9 labels on average.

5.3.2 Implementation Details.

For the CNN component of our model, we use 300 convolution filters with a filter size of 10. We use 300 dimensional word embeddings pretrained on PubMed biomedical article titles and abstracts. To avoid overfitting, we use dropout directly after the embedding layer with a rate of 0.2. For training we use the ADAM (Kingma and Ba, 2014) optimizer with a minibatch size of 8 and a learning rate of 0.001. q, the GCNN hidden layer size, is set to 300.

5.3.3 Evaluation Measures

Thresholding has a large influence on traditional multi-label evaluation measures such as micro-F1 and macro-F1 (Tang et al., 2009). Hence, we report both recall at k (R@k) and precision at k (P@k) which do not require a specific threshold. R@k and P@k is defined in detail in Chapter 2, by Equation 2.5 and Equation 2.4 respectively. R@k is preferred for few- and zero-shot labels, because P@k quickly goes to zero as k increases and gets bigger than the number of group specific labels assigned to each instance. Furthermore, for medical coding, these models are typically used as a recommendation engine to help coders. Unless a label appears at the top of the ranking, the annotator will not see it. Thus, ranking metrics better measure the usefulness of our systems.

5.3.4 Baseline Methods

For the frequent and few-shot labels we compare to state-of-the-art methods on the MIMIC II and MIMIC III datasets including ACNN (Mullenbach et al., 2018) and a CNN method introduced in (Baumel et al., 2017). We also compare with the L1 regularized logistic regression model used in (Vani et al., 2017).

For zero-shot learning, we compare our results with ESZSL (Romera-Paredes and Torr, 2015). To use ESZSL, we must specify feature vectors for each label. For zero-shot methods, the label vectors used are crucial regardless of the learning method used. Therefore, we evaluate ESZSL with three different sets of label vectors. We average 200 dimensional ICD-9 descriptor word embeddings generated by (Pyysalo et al., 2013)[2] which are pretrained on PubMed, Wikipedia, and PubMed Central (ESZSL + W2V). We lowercased descriptors and removed stop-words. We also compare with label vectors derived from our own 300 dimensional embeddings (ESZSL + W2V 2) pretrained on PubMed indexed titles and abstracts. We also generate label vectors using the ICD-9 hierarchy. Specifically, let $\mathbf{Y} \in \mathbb{R}^{N \times L}$ be the document label matrix where N is the total number of documents. We factorize $\mathbf{Y}$ into two matrices $\mathbf{U} \in \mathbb{R}^{N \times 300}$ and $\mathbf{V} \in \mathbb{R}^{300 \times L}$ using graph regularized alternating least squares (GRALS) (Rao et al., 2015). Finally, we also report a baseline using a random ordering on labels, which is important for zero-shot labels — because the total number of such labels is small, the chance that the correct label is in the top k is higher compared to few-shot and frequent labels. This method is explained in detail in Appendix A.

We compare two variants of our method: zero-shot attentive GCNN (ZAGCNN), which is the full method described in Section 5.2 and a simpler variant without the GCNN layers, zero-shot attentive CNN (ZACNN)[3].

5.3.5 Results

Table 5.2 shows the results for MIMIC II. Because the label set for each medical record is augmented using the ICD-9 hierarchy, we expect methods that use the hierarchy to have an advantage. ACNN performs best on frequent labels. For few-shot labels, ZAGCNN outperforms ACNN by over 10% in R@10 and by 8% in R@5; compared to these R@k gains for few-shot labels, our loss on frequent labels is minimal ($< 1\%$). We find that the word embedding derived label vectors work best for ESZSL on zero-shot labels. However, this setup is outperformed by GRALS derived label vectors on the frequent and few-shot labels. On zero-shot labels, ZAGCNN outperforms the best ESZSL variant by over 16% for both R@5 and R@10. Also, we find that the GCNN

---

[2] `http://bio.nlplab.org/`

[3] We name our methods with the "zero-shot" prefix because they are primarily designed for such scenarios, although as we show later that these methods are effective for both few-shot and frequent labels

Table 5.2: MIMIC II results across frequent (S), few-shot (F), and zero-shot (Z) groups. We mark prior methods for MIMIC datasets that we implemented with a *.

	S		F		Z		Harmonic Average	
	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10
Random	00.0	00.0	00.0	00.0	01.1	03.2	–	–
Logistic (Vani et al., 2017) *	13.7	24.7	00.1	00.3	–	–	–	–
CNN (Baumel et al., 2017) *	13.8	25.0	05.0	08.2	–	–	–	–
ACNN (Mullenbach et al., 2018) *	**13.8**	**25.5**	04.6	08.1	–	–	–	–
ESZSL + W2V	07.4	11.9	00.8	01.7	08.0	17.2	02.0	04.1
ESZSL + W2V 2	05.0	08.6	02.5	04.4	10.3	18.9	04.3	07.6
ESZSL + GRALS	13.5	23.8	08.1	12.3	08.5	13.6	09.5	15.2
ZACNN	13.5	24.5	10.3	14.9	14.7	22.1	12.8	20.5
ZAGCNN	13.5	24.7	**13.0**	**18.5**	**26.9**	**36.2**	**16.0**	**24.6**

Table 5.3: MIMIC III results across frequent (S), few-shot (F), and zero-shot (Z) groups. We mark prior methods for MIMIC datasets that we implemented with a *.

	S		F		Z		Harmonic Average	
	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10
Random	00.0	00.0	00.0	00.0	03.8	05.2	–	–
Logistic (Vani et al., 2017) *	27.3	42.7	01.4	01.4	–	–	–	–
CNN (Baumel et al., 2017) *	26.9	41.3	05.8	08.5	–	–	–	–
ACNN (Mullenbach et al., 2018) *	**28.8**	**45.8**	13.0	16.8	–	–	–	–
ESZSL + W2V	13.5	19.1	03.1	05.1	15.7	25.7	06.5	10.5
ESZSL + W2V 2	12.7	18.9	03.1	04.8	14.8	30.5	06.3	10.2
ESZSL + GRALS	25.6	39.3	03.3	06.0	07.6	13.8	06.4	11.4
ZACNN	27.8	43.5	15.2	19.5	36.4	44.2	23.2	31.0
ZAGCNN	28.3	44.5	**16.6**	**21.6**	**42.8**	**49.5**	**25.2**	**33.7**

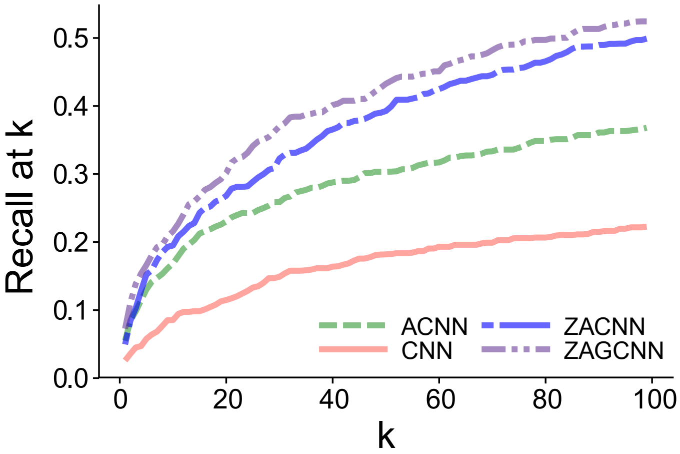
layers help both few- and zero-shot labels. Finally, similar to the setup in (Xian et al., 2017), we also compute the harmonic average across all R@5 and all R@10 scores. The metric is only computed for methods that can predict zero-shot classes. We find that ZAGCNN outperforms ZACNN by 4% for R@10.

We report the MIMIC III results in Table 5.3. Unlike for MIMIC II, the label sets were not expanded using the ICD-9 hierarchy. Yet, we find substantial improvements on both few- and zero-shot labels using a GCNN. ZAGCNN outperforms ACNN by almost 5% and ZACNN by 1% in R@10 on few-shot classes. However, ACNN still outperforms all other methods on frequent labels, but by only 0.3% when compared with ZAGCNN. For zero-shot labels, ZAGCNN outperforms ZACNN by over 5% and outperforms the best ESZSL method by nearly 20% in R@10. We find that ZACNN

Table 5.4: P@k, R@k, and macro-F1 results over all labels (the union of S, F, and Z).

	P@10	R@10	Macro-F1
CNN	56.2	40.7	02.8
ACNN	**62.4**	**45.2**	**06.8**
ZACNN	57.7	42.9	03.7
ZAGCNN	58.7	43.9	03.8

Figure 5.3: This graph plots the MIMIC III R@k for few-shot (F) labels at different k values.



slightly underperforms ZAGCNN on frequent labels with more prominent differences showing up for infrequent labels.

In Table 5.4 we compare the P@10, R@10, and macro-F1 measures across all three groups (the union of S, F, and Z) on the MIMIC III dataset. We emphasize that the evaluation metrics are calculated over all labels and are not averages of the metrics computed independently for each group. We find that R@10 is nearly equivalent to the R@10 on the frequent group in Table 5.3. Furthermore, we find that ACNN outperforms ZAGCNN in P@10 by almost 4%. To compare all methods with respect to macro-F1, we simply threshold each label at 0.5. Both R@k and P@k give more weight to frequent labels, thus it is expected that ACNN outperforms ZAGCNN for frequent labels. However, we also find that ACNN outperforms our methods with respect to Macro-F1.

Given macro-F1 equally weights all labels, does the higher macro score mean

ACNN performs better across infrequent labels? In Figure 5.3, we plot the MIMIC III R@k for the neural methods with k ranging from 1 to 100. We find as k increases, the differences between ZAGCNN and ACNN become more evident. Given Figure 5.3 and the scores in Table 5.3, it is clear that ACNN does not perform better than ZAGCNN with respect to few- and zero-shot labels. The improvement in macro-F1 for ACNN is because it performs better on frequent labels. In general, infrequent labels will have scores much less than 0.5. If we rank all labels ($S \cup F \cup Z$), we find that few-shot labels only occur among the top 16 ranked labels (average number of labels for MIMIC III) for 6% of the test documents that contain them. This result suggests that many frequent irrelevant labels have higher scores than the correct few-shot label.

Why do the rankings among few- and zero-shot labels matter if they are rarely ranked above irrelevant frequent labels? If we can predict which instances contain infrequent labels (novelty detection), then we can help human coders by providing them with multiple recommendation lists — a list of frequent labels and a list of infrequent/zero-shot labels. Also, while we would ideally want a single method that performs best for both frequent and infrequent labels, currently we find that there is a trade-off between them. Hence it may be reasonable to use different methods in combination depending on label frequency.

5.4 Conclusion

In this chapter, we performed a fine-grained evaluation of few- and zero-shot label learning in the large-scale multi-label setting. We also introduced a neural architecture that incorporates label descriptors and the hierarchical structure of the label spaces for few- and zero-shot prediction. For these infrequent labels, previous evaluation methodologies do not provide a clear picture about what works. By evaluating power-law datasets using a generalized zero-shot learning methodology, we provide a staring point toward a better understanding. Our proposed architecture also provides large improvements on infrequent labels over state-of-the-art automatic medical coding methods.

Chapter 6 Conclusion and Future Work

Coding EMRs with diagnosis and procedure codes is an indispensable task for billing, secondary data analyses, and monitoring health trends. Both speed and accuracy of coding are critical. While coding errors could lead to more patient-side financial burden and misinterpretation of a patients well-being, timely coding is also needed to avoid backlogs and additional costs for the healthcare facility. In this thesis, we created several neural network-based methods that overcome various issues encountered when developing automated medical coding systems. In doing so, we provide several contributions to the field with valuable insights for future work. The rest of this chapter discusses our contributions, limitations, and directions for future work in more detail.

6.1 Summary of Contributions

In this dissertation, we presented instantiations of neural networks for coding EMRs with diagnosis and procedure codes. This research has resulted in several important contributions. We list the main contributions below:

Transfer learning for medical coding. In Chapter 3, we performed a detailed analysis of various transfer learning methods to understand what performs best on a real-world EMR dataset. Furthermore, we introduced a simple, yet effective, transfer learning method which overcomes issues of catastrophic forgetting. We show that transfer learning can provide significant improvements to CNNs, especially when using biomedical research articles indexed by the PubMed search engine as supplementary training data.

Matching networks for EMR coding. Traditional CNNs are limited in terms of how well they can perform on power-law datasets, such as the distribution shown in Figure 1.2. In Chapter 4, we introduced a novel extension of matching networks which combine the non-parametric properties of kNN with traditional parametric CNNs. We find that by incorporating the features of similar instances, where the similarity is learned by the neural network, then we can improve over prior state-of-the-art medical coding methods on a variety of evaluation metrics.

Extracting unseen medical codes from EMRs. The shortcoming of many methods for automated medical coding is their inability to predict codes that never occurred in the training dataset. Yet, because many diagnosis and procedure codes may simply be rare, many hospitals may not have encountered them yet. Therefore, many EMR datasets will be missing a lot of codes. However, when a new code is encountered, we still want our models to have the ability to predict them. In Chapter 5, we introduced a model which takes advantage of the structured nature of ICD codes as well as code descriptors to predict unseen codes. Furthermore, we show that if we use a generalized zero-shot learning evaluation methodology, then we can have a clear understanding of how different methods perform on infrequent (few-shot) and unseen (zero-shot) diagnosis and procedure codes.

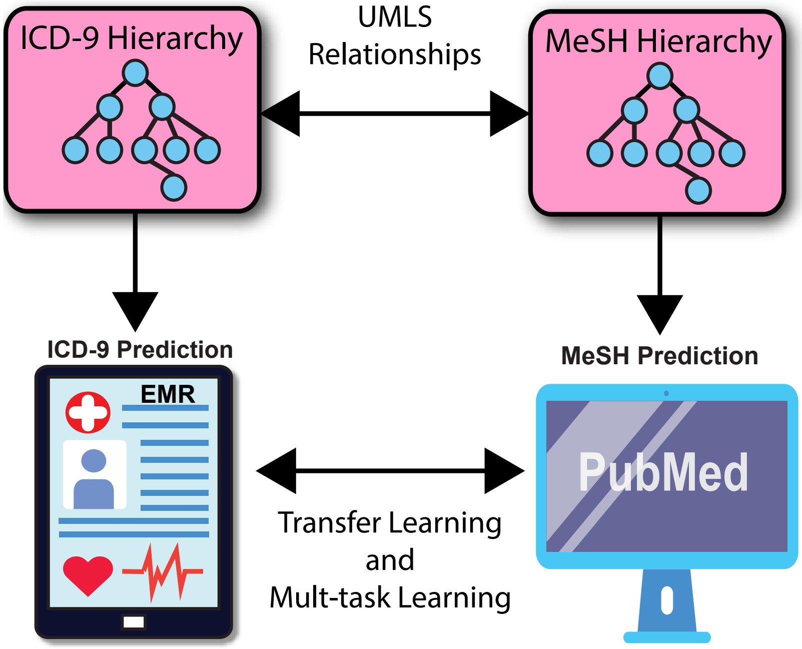
6.2 Limitations and Directions for Future Work

There are limitations with the datasets used to evaluate our various methods. Likewise, there are limitations of the methods presented in this manuscript. Both the MIMIC datsets and the UKY dataset are annotated with procedure and/or diagnosis codes from the ICD-9-CM vocabulary. However, in 2015, a federal mandate was issued that requires healthcare facilities in the United States to use ICD-10 instead of ICD-9. Because of this recent change, ICD-10 training data is limited. Therefore, we use ICD-9 datasets for evaluation. Issues of data sparsity are exacerbated in ICD-10. ICD-9-CM contains 3824 procedure codes and 14025 diagnosis codes. ICD-10-CM contains 71924 procedure codes and 69823 diagnosis codes. So, for a method to extend to ICD-10 codes it must overcome two obstacles. First, with over 140 thousand codes, methods which predict ICD-10-CM codes must scale to such a large label space. Second, with so many labels, it is possible that a greater proportion of codes will be infrequent or not occur at all in many datasets. If an unseen code is not close to seen code in the ICD-10 hierarchy, then our method proposed in Chapter 5 may not be able to predict it. Our semi-parametric matching network also has its own limitations. Specifically, we proposed a simple method to sample support set instances. The method involved repeated distance calculations on a large number training documents. However, as the training set grows, repeatedly performing brute-force distance calculations will become inefficient.

We believe there are three important avenues for future work.

1. For medical coding, a wealth of unstructured domain expertise is available in biomedical research articles indexed by PubMed. These articles are annotated with medical subject headings (MeSH terms), which are organized in a hierarchy. Relationships between MeSH terms and ICD-9 codes are available in

Figure 6.1: Schematic for taking advantage of all the available structured and unstructured information available in PubMed and UMLS.



Unified Medical Language System (UMLS (Bodenreider, 2004)). In Chapter 3, we used transfer learning to try to take advantage of this information. However, if we can take advantage of all this structured and unstructured information via methods such as transfer multi-task learning, then we may be able to predict infrequent labels better. We provide a basic schematic of this idea in Figure 6.1. Furthermore, we only used a relatively small subset of PubMed articles in Chapter 3, if we can develop more efficient training methods, then we can easily take advantage of the 27 million available articles.

2. To predict exact code sets, in Chapters 3 and 4 we rely on simple thresholding methods or a MetaLabeler (Tang et al., 2009). However, as discussed in Chapter 5, these simple thresholding strategies are not sufficient for infrequent and zero-shot codes. A promising area of research is to develop more sophisticated thresholding strategies. Similarly, for our zero-shot medical coding method to be useful for human coders, it is important to develop an accurate novelty detector. We plan to study methods for determining if an instance contains an infrequent label, and if it does, we want to determine how many infrequent labels it should be annotated with. In essence, this is an extension of the MetaLabeler methodology. Novelty detection is also similar to open classification

methods (Shu et al., 2017). Open classification involves dynamic open environments where some new/test documents may not belong to any of the training classes. Open classification methods generally train a extra class which predicts if an instance is not annotated with any of the training classes or not. If we can predict if an instance contains infrequent labels, then we can recommended few- and zero-shot labels to human annotators only when necessary. Likewise, if we can develop better few- and zero-shot methods, then we can use different thresholds for each group in combination with the novelty detector.

3. In this dissertation, we have focused on extracting diagnosis and procedure codes from textual notes in EMRs. The process of coding EMRs with diagnosis and procedure codes can be termed as multi-label classification. In our future work, we can apply our methods to other large multi-label problems besides EMR coding. Other large multi-label problems include classifying Wikipedia articles (Partalas et al., 2015), patent classification (Tran and Kavuluru, 2017), and annotating research articles with MeSH terms (Rios and Kavuluru, 2015a). For example, the matching network described in Chapter 4 and the zero(few)-shot method in Chapter 5 can be used for MeSH classification given PubMed is a power-law dataset where many labels occur infrequently.

Copyright© Anthony Rios, 2018.

Appendix A Graph Regularized Concept Vectors

Zero-shot learning algorithms require a feature vector for each label that we expect to predict at test time. For some applications, attributes are available for each label such that a feature vector can be formed (Xian et al., 2017). When such information is unavailable, then pretrained word vectors of the label names have been used (Xian et al., 2017). In this chapter, we describe GRALS (Rao et al., 2015), a method which we use in Chapter 5 to create label vectors using pairwise relationships between labels. Specifically, we make use the the hierarchical structure of the ICD-9-CM code set. A snippet of the hierarchy is shown in Chapter 5, Figure 5.2.

Model Details In Figure A.1, we display a high-level overview of the GRALS matrix factorization method. Intuitively, we factorize the document label matrix into to smaller matricies: a document matrix, and a label matrix. The label matrix is regularized such that labels that are connected in the hierarchy will have similar vector representations. Unlike Rao et al. (2015), we assume all elements of the document label matrix are observed.

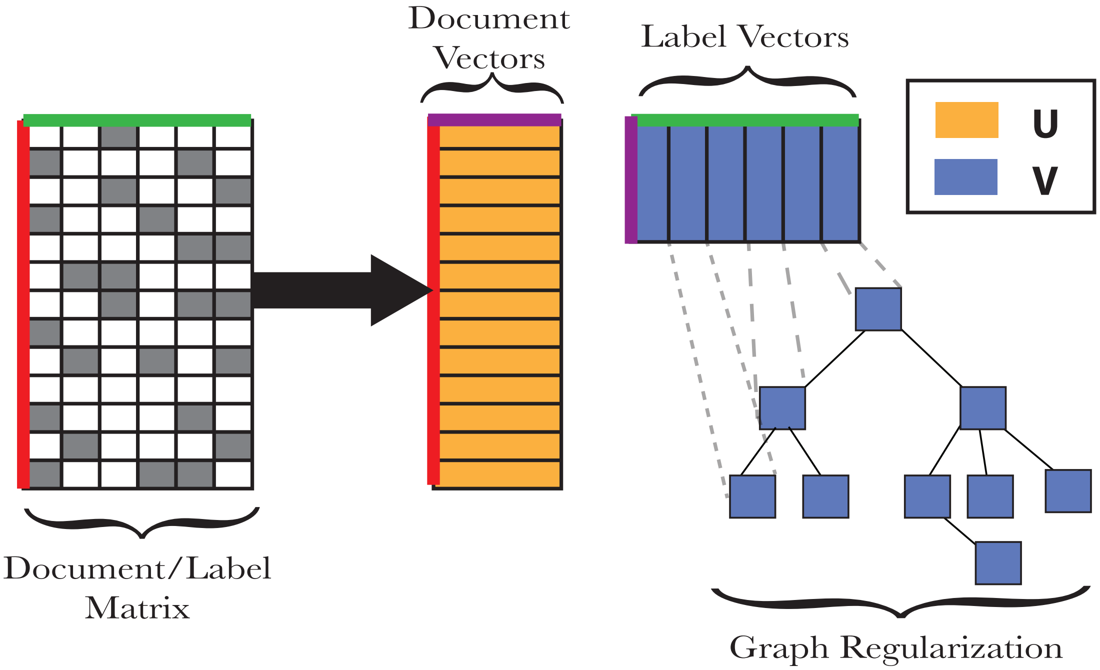
Traditional factorization methods decompose $\mathbf{Y}$ into two matrices, $\mathbf{U} \in \mathbb{R}^{N \times k}$ and $\mathbf{V} \in \mathbb{R}^{L \times k}$ by solvoing the following optimization problem

$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2 + \lambda(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) \tag{A.1}$$

where λ controls the amount of regularization for both $\mathbf{U}$ and $\mathbf{V}$. This factorization has typically been used to reduce the label space such that methods can scale to extreme labels spaces. The main assumption is that $\mathbf{Y}$ can be sufficiently approximated by a low-rank matrix. Several solutions for factoring the document label matrix have been explored by many researchers (Hsu et al., 2009; Yu et al., 2014; Xu et al., 2016). Unfortunately, the low rank approximation of $\mathbf{Y}$ does not hold in the presence of tail labels – infrequently occuring labels. Additionally, if a label never occurs in the training dataset (i.e., the column in $\mathbf{Y}$ for that label is all zeros), then its label vector will be random.

We assume there exists a graph $G = (V, E)$ that encodes structured relationships between labels. If we assume the matrix $\mathbf{V}$ encodes latent label representations, we can assume that labels connected in G will have similar label representations. Even if a ICD-9 code occurs infrequently, if either its parent or child occurs in the training

Figure A.1: Visualization of the GRALS (Rao et al., 2015) matrix factorization method.



dataset, then a label vector for the infrequently occur label can still be reasonably approximated. Therefore, we use the following graph regularizer

$$\sum_{\{i,j\}\in E} E_{i,j}\|\mathbf{v}_i - \mathbf{v}_j\|^2 = tr(\mathbf{V}^T\mathbf{L}_G\mathbf{V}), \tag{A.2}$$

where $E_{i,j}$ is the edge weight between nodes (i, j) and $\mathbf{L}_G = \mathbf{Q} - \mathbf{E}$ is the Laplacian matrix where $\mathbf{Q}$ is the degree matrix of G (Belkin and Niyogi, 2003).

We can combine Equations A.1 and A.2 into the final optimization problem

$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2 + \lambda_1\|\mathbf{U}\|_F^2 + \lambda_2 tr(\mathbf{V}\mathbf{L}_G\mathbf{V}^T). \tag{A.3}$$

To simplify the optimization procedure, and following the work proposed by Rao et al., 2015, we ignore the L2 regularization term on $\mathbf{V}$.

To optimize Equation A.3 we use an alternating minimization technique. First, fixing V and optimizing U, then fixing U and optimizing V.

Optimizing U. To optimize $\mathbf{U}$, we first fix $\mathbf{V}$ which results in the following optimization problem

$$\underset{\mathbf{U}}{\text{minimize}} \quad \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2 + \lambda_1 \|\mathbf{U}\|_F^2 \tag{A.4}$$

which simplifies to standard ridge regression problem. We solve Equation A.4 analytically as

$$\mathbf{U} = \mathbf{Y}\mathbf{V}(\mathbf{V}^T\mathbf{V} + \lambda_1 \mathbf{I})^{-1}$$

where $^{-1}$ represents the inverse of a matrix, and $\mathbf{I}$ is the identity matrix.

Optimizing V. Optimizing $\mathbf{V}$ is equivalent to

$$\underset{\mathbf{V}}{\text{minimize}} \quad \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2 + \lambda_2 tr(\mathbf{V}\mathbf{L}_G\mathbf{V}^T). \tag{A.5}$$

Taking the gradient of $f(\mathbf{V})$ with respect to V and setting it to 0 gives us

$$\mathbf{V}\mathbf{U}^T\mathbf{U} + \lambda_2 \mathbf{L}_G \mathbf{V} = \mathbf{Y}^T\mathbf{U}. \tag{A.6}$$

based on the connections between Frobenius norm and trace functions and their derivatives[1]. Equation A.6 can be solved for $\mathbf{V}$ analytically when $\mathbf{U}$ is fixed. However, it would involve inverting an $L\,k \times L\,k$ matrix which is computationally expensive, especially as the number of codes increases. Therefore, we follow Rao et al. (2015) and use the conjugate gradient (CG) method to find a solution to Equation A.6. In order to perform the CG method efficiently, we need to define an efficient method of calculating a hessian-vector product. Because we use the squared loss, the hessian vector product, $\nabla^2 f(\mathbf{s})\mathbf{s}$, can be calculated as

$$\mathbf{vec}(\mathbf{S}\mathbf{U}^T\mathbf{U} + \lambda_1 \mathbf{S} + \lambda_2 \mathbf{L}_G \mathbf{S})$$

where $\mathbf{vec}(\mathbf{S}) = \mathbf{s}$ is the vectoral representation for the matrix $\mathbf{S}$ obtained by concatenating its columns.

Implementation details. We using an alternating least squares (ALS) optimization procedure to train this model. We repeat the ALS procedure for a total of 25 iterations and perform 10 CG iterations at each step. Both, λ_1 and λ_G are found via grid search and are set to 0.1 and 10.0 respectively. Finally, the vector dimensionality k is set to 300.

---

[1] https://goo.gl/6Cs43b, https://goo.gl/sUUTTE

Abbreviations

ACNN Attention Convolutional Neural Network. 57

ALS Alternating Least Squares. 76

AUCOC Area Under the Receiver Operating Characteristic Curve. 12, 52

AUPRC Area Under the Precision Recall Curve. 12, 52–54

CAC Computer Aided Coding system. 1

CF Convolutional Filter. 25

CG Conjugate Gradient. 76

CMC Computational Medicine Challenge. 9, 14

CNN Convolutional Neural Network. 5, 9, 44, 61

CONSE Convex Combination of Semantic Embeddings. 58

CUIs Concept Unique Identifiers. 23

CV Convolutional Layer. 35

DAG Directed Acyclic Graph. 59

EM Embedding Layer. 35

EMR Electronic Medical Record. 1

ESZSL Embarrassingly Simple Zero-shot Learning. 58, 66

GCNN Graph Convolutional Neural Network. 57

GRALS Graph Regularized Alternating Least Squares. 66, 74

HBR Hierarchical binary relevance. 59

HIPPA Health Insurance Portability and Accountability Act. 1

ICD International Classification of Diseases. 1

ICD-10-CM The International Classification of Diseases, Tenth Revision, Clinical Modification. 1

ICD-9-CM The International Classification of Diseases, Ninth Revision, Clinical Modification. 3, 22, 74

ICU Intensive Care Unit. 51

kNN k Nearest Neighbors. 44

L2R Learning to Rank. 39

LEML Low rank Empirical risk minimization for Multi-Label Learning. 8

MeSH Medical Subject Headings. 31

MIMIC Medical Information Mart for Intensive Care. 1, 50, 63

MNB Multinomial Naive Bayes. 17

nDCG Normalized Discounted Cumulative Gain. 8

NERC Named Entity Recognition. 39

NLM National Library of Medicine. 17

NLP Natural Language Processing. 9

NN Neural Network. 13

P@k Precision at k. 12, 52, 65

R@k Recall at k. 13, 52, 65

RNN Recurrent Neural Network. 10

RTF Rich Text Format. 15

SKR Semantic Knowledge Representation. 16

SVM Support Vector Machine. 10

UKY University of Kentucky. 2

UMLS Unified Medical Language System. 17, 22

W2V Word to Vector (Word2Vec). 66

WHO World Health Organization. 55

XML eXtensible Markup Language. 14

ZACNN Zero-shot Attention Convolutional Neural Network. 66

ZAGCNN Zero-shot Attention Graph Convolutional Neural Network. 66

Bibliography

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. "Label-embedding for image classification". In: *IEEE transactions on pattern analysis and machine intelligence* 38.7 (2016), pp. 1425–1438.

[2] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande. "Low Data Drug Discovery with One-Shot Learning". In: *ACS central science* 3.4 (2017), pp. 283–293.

[3] A. R. Aronson. "Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program." In: *Proceedings of the AMIA Symposium.* American Medical Informatics Association. 2001, p. 17.

[4] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Simaan. "Graph Convolutional Encoders for Syntax-aware Neural Machine Translation". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP).* 2017, pp. 1957–1967.

[5] T. Baumel, J. Nassour-Kassis, M. Elhadad, and N. Elhadad. "Multi-Label Classification of Patient Notes a Case Study on ICD Code Assignment". In: *arXiv preprint arXiv:1709.09587* (2017).

[6] M. Belkin and P. Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. "A neural probabilistic language model". In: *The Journal of Machine Learning Research* 3 (2003), pp. 1137–1155.

[8] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. "Sparse local embeddings for extreme multi-label classification". In: *Advances in neural information processing systems (NIPS).* 2015, pp. 730–738.

[9] W. Bi and J. T. Kwok. "Multi-label classification on tree-and DAG-structured hierarchies". In: *International Conference on Machine Learning (ICML).* 2011, pp. 17–24.

[10] O. Bodenreider, S. Nelson, W. Hole, and H. Chang. "Beyond synonymy: exploiting the UMLS semantics in mapping vocabularies". In: *Proceedings of AMIA Symposium.* American Medical Informatics Association, 1998, pp. 815–819.

[11] O. Bodenreider. "The unified medical language system (UMLS): integrating biomedical terminology". In: *Nucleic acids research* 32.suppl_1 (2004), pp. D267–D270.

[12] A. Bordes, N. Usunier, S. Chopra, and J. Weston. "Large-scale simple question answering with memory networks". In: *arXiv preprint arXiv:1506.02075* (2015).

[13] A. C. P. L. F. de Carvalho and A. A. Freitas. "A Tutorial on Multi-label Classification Techniques". In: *Foundations of Computational Intelligence (5)*. Vol. 205. 2009, pp. 177–195.

[14] C.-C. Chang and C.-J. Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm` (Accessed: April 8, 2015), 27:1–27:27.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[16] M. Chen, Z. Lin, and K. Cho. "Graph Convolutional Networks for Classification with a Structured Label Space". In: *arXiv preprint arXiv:1710.04908* (2017).

[17] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun. "Doctor ai: Predicting clinical events via recurrent neural networks". In: *Machine Learning for Healthcare Conference*. 2016, pp. 301–318.

[18] R. Collobert and J. Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *International Conference on Machine learning (ICML)*. ACM. 2008, pp. 160–167.

[19] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. "Natural Language Processing (Almost) from Scratch". In: *Journal of Machine Learning Research* 12 (2011), pp. 2493–2537.

[20] M. Defferrard, X. Bresson, and P. Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 3844–3852.

[21] M. Dougherty, S. Seabold, and S. E. White. "Study reveals hard facts on CAC". In: *Journal of the American Health Information Management Association* 84.7 (2013), pp. 54–56.

[22] F. Duarte, B. Martins, C. S. Pinto, and M. J. Silva. “A Deep Learning Method for ICD-10 Coding of Free-Text Death Certificates”. In: *Portuguese Conference on Artificial Intelligence*. Springer. 2017, pp. 137–149.

[23] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. “LIBLINEAR: A Library for Large Linear Classification”. In: *Journal of Machine Learning Research* 9 (2008), pp. 1871–1874. ISSN: 1532-4435.

[24] J. Y. Foo, G. K. Davis, and M. A. Brown. “Frontal lobe meningioma mimicking preeclampsia: A case study”. In: *Obstetric medicine* 10.4 (2017), pp. 192–194.

[25] G. Forman. “An extensive empirical study of feature selection metrics for text classification”. In: *Journal of Machine Learning Research* 3 (Mar. 2003), pp. 1289–1305. ISSN: 1532-4435.

[26] J. Fürnkranz, E. Hüllermeier, E. Loza Menca, and K. Brinker. “Multilabel classification via calibrated label ranking”. In: *Machine Learning* 73.2 (Nov. 2008), pp. 133–153. ISSN: 0885-6125.

[27] D. Gentile. “Pathological video-game use among youth ages 8 to 18: a national study”. In: *Psychological science* 20.5 (2009), pp. 594–602.

[28] X. Glorot, A. Bordes, and Y. Bengio. “Deep sparse rectifier networks”. In: *International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*. Vol. 15. 2011, pp. 315–323.

[29] M. L. Gundersen, P. J. Haug, T. A. Pryor, R. van Bree, S. Koehler, K. Bauer, and B. Clemons. “Development and evaluation of a computerized admission diagnosis encoding system”. In: *Computers and Biomedical Research* 29.5 (Oct. 1996), pp. 351–372. ISSN: 0010-4809.

[30] H. He and E. A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (Sept. 2009), pp. 1263–1284. ISSN: 1041-4347.

[31] J. Howard and S. Ruder. “Fine-tuned Language Models for Text Classification”. In: *arXiv preprint arXiv:1801.06146* (2018).

[32] D. J. Hsu, S. M. Kakade, J. Langford, and T. Zhang. “Multi-label prediction via compressed sensing”. In: *Advances in neural information processing systems (NIPS)*. 2009, pp. 772–780.

[33] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. “Snapshot ensembles: Train 1, get m for free”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[34] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. "Deep unordered composition rivals syntactic methods for text classification". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015, pp. 1681–1691.

[35] H. Jain, Y. Prabhu, and M. Varma. "Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications". In: *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM. 2016, pp. 935–944.

[36] A. Johnson, T. Pollard, L. Shen, L. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Celi, and R. Mark. "MIMIC-III, a freely accessible critical care database". In: *Scientific data* 3 (2016).

[37] R. Johnson and T. Zhang. "Deep pyramid convolutional neural networks for text categorization". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. Vol. 1. 2017, pp. 562–570.

[38] V. Jouhet, G. Defossez, A. Burgun, P. le Beux, P. Levillain, P. Ingrand, and V. Claveau. "Automated classification of free-text pathology reports for registration of incident cases of cancer". In: *Methods of Information in Medicine* 51.3 (2012), pp. 242–251.

[39] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. "A Convolutional Neural Network for Modeling Sentences". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2014, pp. 655–665.

[40] N. Kamra, U. Gupta, and Y. Liu. "Deep Generative Dual Memory Network for Continual Learning". In: *arXiv preprint arXiv:1710.10368* (2017).

[41] R. Kavuluru and A. Rios. "Automatic Assignment of Non-Leaf MeSH Terms to Biomedical Articles". In: *AMIA 2015, American Medical Informatics Association Annual Symposium*. 2015.

[42] R. Kavuluru, A. Rios, and Y. Lu. "An empirical evaluation of supervised learning approaches in assigning diagnosis codes to electronic medical records". In: *Artificial intelligence in medicine* 65.2 (2015), pp. 155–166.

[43] Y. Kim. "Convolutional Neural Networks for Sentence Classification". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1746–1751.

[44] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. 2014.

[45] T. N. Kipf and M. Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[46] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3521–3526.

[47] G. Koch, R. Zemel, and R. Salakhutdinov. “Siamese neural networks for one-shot image recognition”. In: *ICML Deep Learning Workshop*. Vol. 2. 2015.

[48] J. Lee, D. J. Scott, M. Villarroel, G. D. Clifford, M. Saeed, and R. G. Mark. “Open-access MIMIC-II database for intensive care research”. In: *IEEE Annual International Conference Engineering in Medicine and Biology Society (EMBC)*. 2011, pp. 8315–8318.

[49] Z. Li and D. Hoiem. “Learning without forgetting”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[50] L. R. S. de Lima, A. H. F. Laender, and B. A. Ribeiro-Neto. “A hierarchical approach to the automatic categorization of medical documents”. In: *International Conference on Information Management (CIKM)*. 1998, pp. 132–139.

[51] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. “Automatic discovery and optimization of parts for image classification”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[52] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang. “Deep Learning for Extreme Multi-label Text Classification”. In: *International Conference on Research and Development in Information Retrieval (SIGIR)*. 2017, pp. 115–124.

[53] D. Marcheggiani and I. Titov. “Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017, pp. 1506–1515.

[54] T. Mensink, E. Gavves, and C. G. Snoek. “Costa: Co-occurrence statistics for zero-shot classification”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 2441–2448.

[55] G. Meyer, C. Denham, J. Battles, et al. “Safe practices for better healthcare–2010 update: a consensus report”. In: *Washington, DC, National Quality Forum*. 2010, p. 58.

[56] T. Mikolov, A. Frome, S. Bengio, J. Shlens, Y. Singer, G. S. Corrado, J. Dean, and M. Norouzi. “Zero-shot learning by convex combination of semantic embeddings”. In: *International Conference on Learning Representations (ICLR)*. 2013.

[57] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed representations of words and phrases and their compositionality”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 3111–3119.

[58] A. H. Miller, A. Fisch, J. Dodge, A. Karimi, A. Bordes, and J. Weston. “Key-Value Memory Networks for Directly Reading Documents”. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2016, pp. 1400–1409.

[59] L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang, and Z. Jin. “How Transferable are Neural Networks in NLP Applications?” In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2016, pp. 479–489.

[60] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein. “Explainable Prediction of Medical Codes from Clinical Text”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[61] V. Nair and G. E. Hinton. “Rectified linear units improve restricted Boltzmann machines”. In: *International Conference on Machine Learning (ICML)*. 2010, pp. 807–814.

[62] J. Nam, J. Kim, E. Loza Mencía, I. Gurevych, and J. Fürnkranz. “Large-Scale Multi-label Text Classification - Revisiting Neural Networks”. In: *European Conference on Machine Learning and Knowledge Discovery in Databases - (ECML PKDD)*. 2014, pp. 437–452.

[63] J. Nam, J. Kim, E. L. Menca, I. Gurevych, and J. Fürnkranz. “Large-scale Multi-label Text ClassificationRevisiting Neural Networks”. In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 437–452.

[64] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng. “On optimization methods for deep learning”. In: *International Conference on Machine Learning (ICML)*. 2011, pp. 265–272.

[65] J. S. Olsson. “Combining feature selectors for text classification”. In: *Proc. the 15th ACM international conference on Information and knowledge management*. 2006, pp. 798–799.

[66] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. “Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1717–1724.

[67] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artières, G. Paliouras, É. Gaussier, I. Androutsopoulos, M. Amini, and P. Gallinari. “LSHTC: A Benchmark for Large-Scale Text Classification”. In: *CoRR* abs/1503.08581 (2015).

[68] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang. “Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN”. In: *World Wide Web Conference on World Wide Web*. 2018, pp. 1063–1072.

[69] A. Perotte, R. Pivovarov, K. Natarajan, N. Weiskopf, F. Wood, and N. Elhadad. “Diagnosis code assignment: models and evaluation metrics”. In: *Journal of the American Medical Informatics Association* 21.2 (2013), pp. 231–237.

[70] J. P. Pestian, C. Brew, D. J. Matykiewicz Paweand Hovermale, N. Johnson, K. B. Cohen, and W. Duch. “A shared task involving multi-label classification of clinical free text”. In: *Proceedings of the Workshop on BioNLP: Biological, Translational, and Clinical Language Processing*. 2007, pp. 97–104.

[71] Y. Prabhu and M. Varma. “Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning”. In: *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM. 2014, pp. 263–272.

[72] A. Prakash, S. Zhao, S. A. Hasan, V. V. Datla, K. Lee, A. Qadir, J. Liu, and O. Farri. “Condensed Memory Networks for Clinical Diagnostic Inferencing.” In: *AAAI*. 2017, pp. 3274–3280.

[73] S. Pyysalo, F. Ginter, H. Moen, T. Salakoski, and S. Ananiadou. “Distributional Semantics Resources for Biomedical Text Processing”. In: *LBM*. 2013, pp. 39–44.

[74] J. Quevedo, O. Luaces, and A. Bahamonde. “Multilabel classifiers with a probabilistic thresholding strategy”. In: *Pattern Recognition* 45.2 (2012), pp. 876–883. ISSN: 0031-3203.

[75] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon. “Collaborative filtering with graph information: Consistency and scalable methods”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 2107–2115.

[76] J. Read, B. Pfahringer, and G. Holmes. “Multi-label Classification Using Ensembles of Pruned Sets”. In: *IEEE International Conference on Data Mining (ICDM)*. 2008, pp. 995–1000.

[77] J. Read, B. Pfahringer, G. Holmes, and E. Frank. “Classifier Chains for Multi-label Classification”. In: *Machine Learning* 85.3 (2011), pp. 335–359.

[78] T. C. Rindflesch, B. Libbus, D. Hristovski, A. R. Aronson, and H. Kilicoglu. “Semantic relations asserting the etiology of genetic diseases”. In: *AMIA Annual Symposium Proceedings*. Vol. 2003. American Medical Informatics Association. 2003, p. 554.

[79] A. Rios and R. Kavuluru. “Analyzing the Moving Parts of a Large-Scale Multi-Label Text Classification Pipeline: Experiences in Indexing Biomedical Articles.” In: *IEEE International Conference on Healthcare Informatics. IEEE International Conference on Healthcare Informatics*. Vol. 2015. 2015, pp. 1–7.

[80] A. Rios and R. Kavuluru. “Convolutional neural networks for biomedical text classification: application in indexing biomedical articles”. In: *Conference on Bioinformatics, Computational Biology and Health Informatics, (BCB)*. 2015, pp. 258–267.

[81] A. Rios and R. Kavuluru. “EMR Coding with Semi-Parametric Multi-Head Matching Networks”. In: *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[82] A. Rios and R. Kavuluru. “Ordinal convolutional neural networks for predicting RDoC positive valence psychiatric symptom severity scores”. In: *Journal of biomedical informatics* 75 (2017), S85–S93.

[83] A. Rios and R. Kavuluru. “Supervised extraction of diagnosis codes from EMRs: role of feature selection, data selection, and probabilistic thresholding”. In: *IEEE International Conference on Healthcare Informatics (ICHI)*. 2013, pp. 66–73.

[84] A. Rios, R. Kavuluru, and Z. Lu. “Generalizing Biomedical Relation Classification with Neural Adversarial Domain Adaptation”. In: *Bioinformatics* 1 (2018), p. 9.

[85] A. Rios, R. Vanderpool, P. Shaw, and R. Kavuluru. “A Multi-Label Classification Approach for Coding Cancer Information Service Chat Transcripts”. In: *The Twenty-Sixth International FLAIRS Conference*. 2013.

[86] B. Romera-Paredes and P. Torr. "An embarrassingly simple approach to zero-shot learning". In: *International Conference on Machine Learning*. 2015, pp. 2152–2161.

[87] T. N. Rubin, A. Chambers, P. Smyth, and M. Steyvers. "Statistical topic models for multi-label document classification". In: *Machine Learning* 88.1-2 (2012), pp. 157–208.

[88] S. K. Sahu and A. Anand. "What matters in a transferable neural network model for relation classification in the biomedical domain?" In: *arXiv preprint arXiv:1708.03446* (2017).

[89] U. Sandouk and K. Chen. "Multi-label zero-shot learning via concept embedding". In: *arXiv preprint arXiv:1606.00282* (2016).

[90] C. dos Santos, B. Xiang, and B. Zhou. "Classifying Relations by Ranking with Convolutional Neural Networks". In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015, pp. 626–634.

[91] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling. "Modeling Relational Data with Graph Convolutional Networks". In: *Proceedings of 15th European Semantic Web Conference (ESWC 2018)*. 2018.

[92] J. Shreve, J. Van, D. Bos, T. Gray, M. Halford, K. Restagi, and E. Ziemkiwicz. "The Economic Measurement of Medical Errors". In: *Society of Actuaries* (2010).

[93] L. Shu, H. Xu, and B. Liu. "DOC: Deep Open Classification of Text Documents". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2017, pp. 2911–2916.

[94] J. Snell, K. Swersky, and R. Zemel. "Prototypical Networks for Few-shot Learning". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 4078–4088.

[95] J. Snell, K. Swersky, and R. Zemel. "Prototypical networks for few-shot learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4080–4090.

[96] R. Socher. "Recursive deep learning for natural language processing and computer vision". PhD thesis. Citeseer, 2014.

[97] S. Sohn, W. Kim, D. C. Comeau, and W. J. Wilbur. “Optimal Training Sets for Bayesian Prediction of MeSH Term Assignment.” In: *Journal of the American Medical Informatics Association* 15.4 (2008), pp. 546–553.

[98] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: A simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[99] S. Al-Stouhi and C. K. Reddy. “Transfer learning for class imbalance problems with inadequate data”. In: *Knowledge and information systems* 48.1 (2016), pp. 201–228.

[100] E. Strubell and A. McCallum. “Dependency Parsing with Dilated Iterated Graph CNNs”. In: *Workshop on Structured Prediction for Natural Language Processing, SPNLP@EMNLP*. 2017, pp. 1–6.

[101] S. Sukhbaatar, J. Weston, R. Fergus, et al. “End-to-end memory networks”. In: *Advances in neural information processing systems*. 2015, pp. 2440–2448.

[102] F. Tai and H.-T. Lin. “Multilabel classification with principal label space transformation”. In: *Neural Computation* 24.9 (2012), pp. 2508–2542.

[103] L. Tang, S. Rajan, and V. K. Narayanan. “Large scale multi-label classification via metalabeler”. In: *Proceedings of the 18th international conference on World wide web*. ACM. 2009, pp. 211–220.

[104] T. Tran and R. Kavuluru. “Supervised Approaches to Assign Cooperative Patent Classification (CPC) Codes to Patents”. In: *International Conference on Mining Intelligence and Knowledge Exploration*. Springer. 2017, pp. 22–34.

[105] G. Tsoumakas, I. Katakis, and I. P. Vlahavas. “Mining Multi-label Data”. In: *Data Mining and Knowledge Discovery Handbook*. 2010, pp. 667–685.

[106] A. Vani, Y. Jernite, and D. Sontag. “Grounded Recurrent Neural Networks”. In: *arXiv preprint arXiv:1705.08557* (2017).

[107] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. “Matching networks for one shot learning”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[108] B. C. Wallace, K. Small, C. E. Brodley, and T. A. Trikalinos. “Class imbalance, redux”. In: *IEEE International Conference on Data Mining (ICDM)*. 2011, pp. 754–763.

[109] X. Wang, Y. Ye, and A. Gupta. "Zero-shot Recognition via Semantic Embeddings and Knowledge Graphs". In: *arXiv preprint arXiv:1803.08035* (2018).

[110] J. Weston, S. Chopra, and A. Bordes. "Memory networks". In: *arXiv preprint arXiv:1410.3916* (2014).

[111] J. Wiens, J. Guttag, and E. Horvitz. "A study in transfer learning: leveraging data from multiple hospitals to enhance hospital-specific predictions". In: *Journal of the American Medical Informatics Association* 21.4 (2014), pp. 699–706.

[112] R. T. Wood. "Problems with the concept of video game addiction: Some case study examples". In: *International Journal of Mental Health and Addiction* 6.2 (2008), pp. 169–178.

[113] B. Wu, S. Lyu, and B. Ghanem. "Ml-mg: Multi-label learning with missing labels using a mixed graph". In: *IEEE International Conference on Computer Vision (ICCV).* 2015, pp. 4157–4165.

[114] Y. Xian, B. Schiele, and Z. Akata. "Zero-shot learning-The Good, the Bad and the Ugly". In: *IEEE Conference on Computer Vision and Pattern Recognition.* IEEE Computer Society. 2017, pp. 3077–3086.

[115] C. Xu, D. Tao, and C. Xu. "Robust Extreme Multi-label Learning." In: *KDD.* 2016, pp. 1275–1284.

[116] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. "Hierarchical attention networks for document classification". In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 2016, pp. 1480–1489.

[117] H. Yu, P. Jain, P. Kar, and I. S. Dhillon. "Large-scale Multi-label Learning with Missing Labels". In: *International Conference on Machine Learning (ICML).* 2014, pp. 593–601.

[118] M. D. Zeiler. "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[119] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. "Deconvolutional networks". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE. 2010, pp. 2528–2535.

[120] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, et al. "Relation Classification via Convolutional Deep Neural Network." In: *COLING.* 2014, pp. 2335–2344.

[121] D. Zhang, D. He, S. Zhao, and L. Li. “Enhancing Automatic ICD-9-CM Code Assignment for Medical Texts with PubMed”. In: *BioNLP 2017* (2017), pp. 263–271.

[122] M.-L. Zhang and K. Zhang. “Multi-label learning by exploiting label dependency”. In: *Proceedings of the 16th ACM SIGKDD*. KDD ’10. 2010, pp. 999–1008. ISBN: 978-1-4503-0055-1.

Vita

Name

Anthony Rios

Education

- 2007–2011 B.S. in Computer Science GEORGETOWN COLLEGE Georgetown, Kentucky

Experience

- 2017, Summer Research Fellow, National Institute of Health (NCBI/NLM/NIH), Bethesda, Maryland
- 2013–present, Graduate Research Assistant, University of Kentucky, Lexington, Kentucky
- 2010–2013, Software Engineer Intern/Co-op, Lexmark International, Lexington, Kentucky
- Spring 2010, Software Engineer Intern, Corevalus Systems LLC., Georgetown, Kentucky

Awards

- 2017 – Best poster, Annual Commonwealth Computational Summit
- 2017 – Ranked 1st (among 13 teams; 500 Euro prize) in the BioCreative CHEMPROT text mining shared task
- 2017 – Ranked 2nd (among 11 teams) in the shared task on classification of medication intake messages on Twitter for online pharmacovigilance (at Social media mining for health workshop at AMIA)
- 2017 – NIH Intramural Research Training Award (IRTA)
- 2016 – Ranked 3rd (among 24 teams) in the CEGS NGRID shared task on predicting psychiatric symptom severity scores based on clinical notes (RDoC for Psychiatry workshop at AMIA)

- 2015 – Thaddeus B. Curtz Memorial Scholarship, University of Kentucky
- 2015 – Best paper finalist, IEEE ICHI 2015.
- 2015 – Ranked 2nd (among 18 teams), Annual BioASQ Semantic Indexing Challenge, Task A (Batch 2)
- 2014 – Distinguished poster finalist, AMIA
- 2011 – Outstanding Senior in Computer Science, Georgetown College

Publications

1. Y. Peng, **A. Rios**, R. Kavuluru, and Z. Lu. Extracting chemical-protein relations with ensembles of SVM and deep learning models. Database. 2018.
2. **A. Rios**, T. Tran, and R. Kavuluru. Predicting Psychological Health from Childhood Essays with Convolutional Neural Networks for the CLPsych 2018 Shared Task (Team UKNLP). In Fifth Annual Computational Linguistics and Clinical Psychology Workshop (CLPsych). 2018
3. **A. Rios**, R. Kavuluru, and Z. Lu. Generalizing Biomedical Relation Classification with Neural Adversarial Domain Adaptation. Bioinformatics. 2018.
4. **A. Rios** and R. Kavuluru. EMR Coding with Semi-Parametric Multi-Head Matching Networks. In Conference of the North American Chapter of the Association for Computational Linguistics (NAACL); 2018.
5. **A. Rios** and R. Kavuluru, Ordinal Convolutional Neural Networks for Predicting RDoC Positive Valence Psychiatric Symptom Severity Scores. Journal of Biomedical Informatics, Volume 75, Pages S85-S93, 2017.
6. Y. Peng, **A. Rios**, R. Kavuluru, and Z. Lu. Chemical-protein relation extraction with SVM, CNN, RNN and ensemble systems. In 6th BioCreative Challenge Evaluation Workshop. 2017
7. S. Han, T. Tran, **A. Rios**, and R. Kavuluru. Team UKNLP: Detecting ADRs, Classifying Medication In-take Messages, and Normalizing ADR Mentions on Twitter. In 2nd Social Media Mining for Health Applications Workshop and Shared Task at AMIA. 2017
8. R. Kavuluru, **A. Rios**, and T. Tran. Extracting Drug-Drug Interactions with Word and Character-Level Recurrent Neural Networks. In IEEE International Conference on Healthcare Informatics, Workshop on Healthcare Knowledge Discovery and Management. 2017

9. R. Kavuluru, **A. Rios**, and Y. Lu. An Empirical Evaluation of Supervised Learning Approaches in Assigning Diagnosis Codes to Electronic Medical Records. Artificial Intelligence in Medicine, Volume 65, Issue 2; 2015.
10. R. Kavuluru and **A. Rios**. Automatic Assignment of Non-Leaf Medical Subject Headings to Biomedical Articles. Proceedings of the American Medical Informatics Association annual symposium; 2015
11. **A. Rios** and R. Kavuluru. Analyzing the Moving Parts of a Large-Scale Multi-Label Text Classification Pipeline: Experiences in Indexing Biomedical Articles. In IEEE International Conference on Healthcare Informatics; 2015
12. **A. Rios** and R. Kavuluru. Convolutional Neural Networks for Biomedical Text Classification: Application in Indexing Biomedical Articles. In 6th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics; 2015
13. R. Kavuluru, **A. Rios**, Brandon Kulengowski, and Patrick McNamara. A Knowledge-Based Collaborative Clinical Case Mining Framework. In American Medical Informatics Association (AMIA) annual symposium; 2014
14. **A. Rios** and R. Kavuluru. Supervised Extraction of Diagnosis Codes from EMRs: Role of Feature Selection, Data Selection, and Probabilistic Thresholding. In IEEE International Conference on Healthcare Informatics; 2013
15. **A. Rios**, R. Vanderpool, P. Shaw, and R. Kavuluru. A Multi-Label Classification Approach to Coding Cancer Information Service Chat Transcripts. In 26th International Florida AI Research Society conference; 2013