NLP for healthcare Final Report

https://youtu.be/1aYPVbVvgC0

Anastasios Stathopoulos, Luis Manuel Garcia-Baquero Corredera, Ashwin Malgaonkar, Sagar Jounkani

Georgia Institute of Technology

Abstract

Clinical notes are created all the time for every patient by clinicians and they include valuable information like diagnosis and medication treatment. They are always annotated by medical codes indicating the diagnosis and treatment applied. This process takes a lot of labor time since it requires analytical reading of the document from the medical staff and at the same time very good knowledge of the vast number of ICD codes. Every document may be tagged with many medical ICD codes increasing the time it takes to be annotated properly. In order to reduce this time a new approach will be implemented in this project as introduced in the paper (James Mullenbach, 2018) which fully automates the labelling process of a document of clinical notes. The expected result would be the dramatical reduction of the time the labelling of such a document takes leaving for the clinicians only the validating process and at the same time the increase of reliability and accuracy of labelling.

I. Introduction

Clinical notes are free text documents created by clinicians during patient encounters. They are typically tagged by medical codes indicating the diagnosis and treatment applied. Labelling of clinical notes has been proved to be a very difficult task. First of all, the vast numbers of ICD-9 codes increase dramatically the dimensionality of the problem making it error-prone. Secondly useless information along with custom abbreviations may lead to wrong labelling even for humans. Tagging clinical notes is very important as it classifies the treatment and diagnosis to ICD-9 and ICD-10 codes providing valuable information to the medical community. Automation of this process has been on pursuit for the past years with NLP be the main utility of these attempts. With the rise of neural networks (NN), a combination of NN architecture and NLP gave more promising results. Convolutional Neural Network (CNN) architectures along with RNN (Recurrent Neural Network) architectures are among the top solutions so far (T. Kang, 2017). The concept of this approach is to import a document and convert it to a base representation (word2vec) (T. Mikolov, 2013) so it can then be classified and annotated with the correct labels. Our deep learning model will analyze the document representation and try to annotate it with relevant codes. The basic idea behind this approach is that important information is hidden in small snippets of text and for every label they are different. This is the main reason the author is using a convolutional neural network along with an attention mechanism.

II. Methods

This problem can be characterized as a multi-label classification problem. If L is a subset of ICD-9 codes, we want to use a deep learning model to make an L binary classification decision system.

Data Preprocessing

For this project we use the MIMIC-III dataset. The whole pipeline is implemented with a big data framework: Spark using Scala. The preprocessing steps are:

1) Load data.
    a. Use the discharge summaries as raw documents from the table: NOTEEVENTS.csv
    b. Use the DIAGNOSES_ICD table to extract ICD-9 codes per patient per attendance
    c. Use the PROCEDURES_ICD table to extract ICD-9 codes per patient per attendance

2) Drop rows with empty values from DIAGNOSES_ICD-9 that have empty values.
3) Convert codes into ICD-9 format. Procedures and Diagnoses codes follows a certain format with dots.
4) Filter discharge summaries by removing special characters like (,),*,_ and remove numeric only words (drop "234" but keep "200grams")
5) Tokenize every summary so it is transformed to an array of words
6) Filter out entries whose admissions do not have discharge summaries. It is noted that for some admissions there are no discharge summaries, so we filter them out both from notes and codes entries.
7) Truncate number of tokens per document to a maximum of 2500 tokens
8) Aggregate data and construct a unique dataset (Discharge summaries and icd-9 codes)
9) Split data into train, validation, test sets with a percentage of 80%, 10%,10% respectively and make sure no patient id admissions is shared among these sets, so no common patient id is identified in either of those sets.
10) Create our vocabulary. The vocabulary is derived from the unique words they are identified in the entire dataset of discharge summaries. We drop words that appear less than 3 times in the entire dataset.
11) Use Word2Vec to train embedding weights for out vocabulary

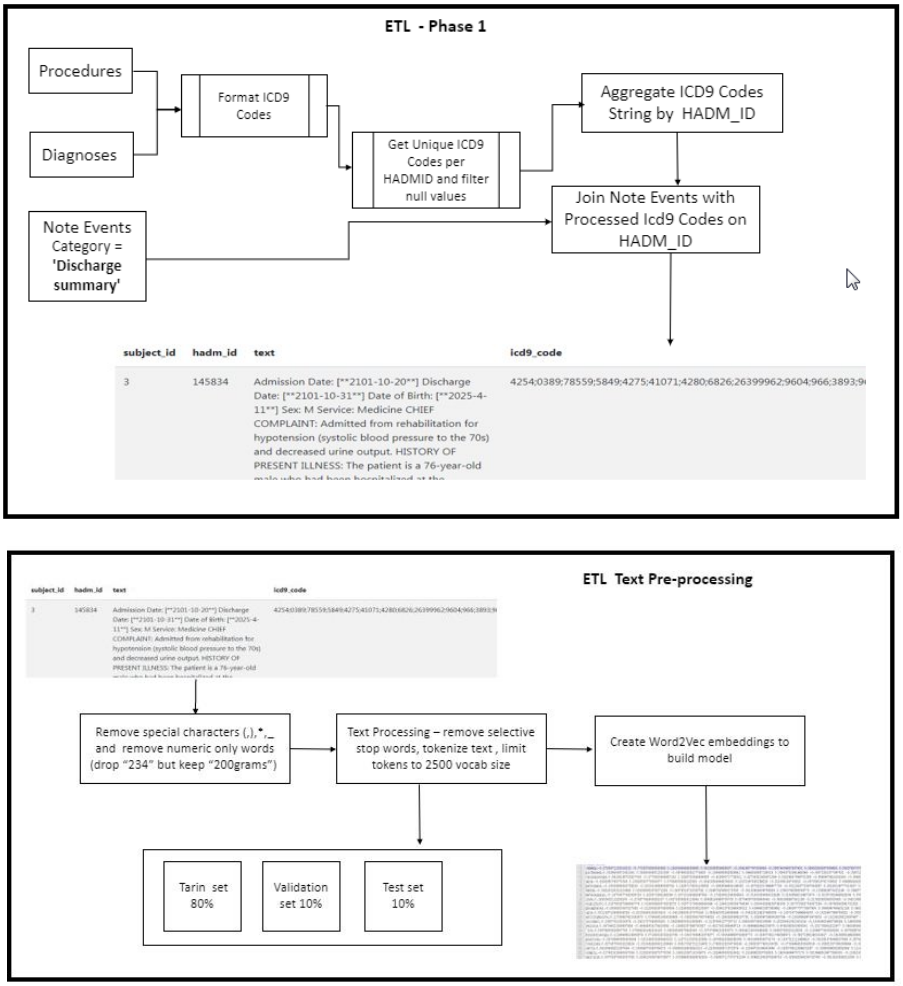

Figure 1: First phase of pre-processing steps (top) and second phase (bottom)

During the draft phase we experienced very bad results since the dataset was heavily imbalanced and few labels dominated making the system very optimistic about them and pessimistic about rare labels. Also during training, the loss function after 1rst epoch dropped significantly and stayed the same throughout the training. This confirms our assumptions of class imbalance. Following the approach of many papers (James Mullenbach, 2018), we decided to use samples that contain only the top 50 most used labels. This approach proved to be very good since from the initial

set of 52722 samples we ended up with 49617 samples. As displayed in the Figure 2 the new dataset is much more balanced than before.

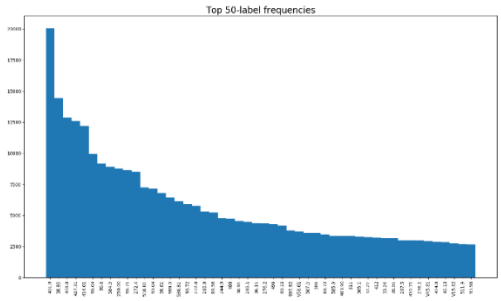

Figure 2: top 50 class distribution

Feature Extraction

In order to feed our data into our model we need to create features from out preprocessed data. For this reason, we convert the array of words (tokens) for every summary to a 2D array of vectors using the embedding technic (Tomas Mikolov, 2013). For every word we have a 1D vector representation of d- depth. We chose 100 vector size. Then, we create a dictionary for every word by assigning an increasing ID as a key and we do the same for labels. The end result of the summary representation is a num_tokens X d matrix where num_tokens represents the number of tokens of the summary and the d is the depth of the embeddings (in this case 100). In order to deal with different summary length, we utilize a padding method for every batch. We find out the largest length of token array in the batch and we pad with zeros the rest of the tokenized rows. For the zeros elements we assign an embedding vector of zeros and for the unknown words that do not exist in the vocabulary we create random vectors of size 100.

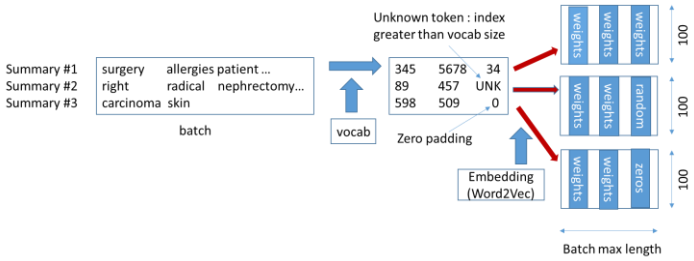

Figure 3: Feature engineering. Input: tokenized docs where every row represents a document. Output: 3D matrix of size: (batch_size x batch max length x 100)

Model Establishment

The input of the model is a list of indices corresponding to the indices of the vocabulary created from the word2vec training phase. The output of the model does not include a softmax function since for our loss we are using the *BCEWithLogitsLoss*() which applies automatically the softmax function converting scores to probabilities. We consider as a probability threshold the value 0.5. If output of class i is greater than 0.5 we consider that class is active and after we find the label text from the dictionary we assign this label to the document. The function loss we are using is the Binary cross entropy loss given below:

$loss = -\sum_{i=1}^{N}[y_i log p_i + (1 - y_i)\log(1 - p_i)]$, where N the number of categories, y_i the binary indicator for category i, p_i predicted probability for category i. The optimizer selected for the model is Adam optimizer, a sophisticated optimizer used extensively is such problems.

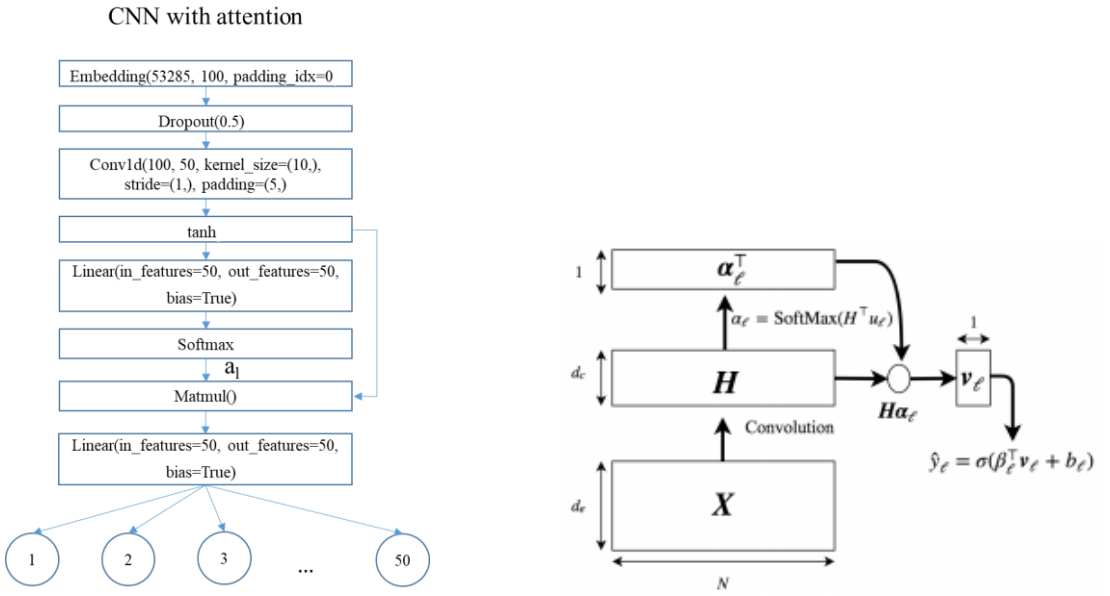

Figure 4: CNN with attn model structure (left). The high level design of the model as described in (James Mullenbach, 2018)

For the sake of simplicity we do not illustrate the architectures of the other models used for comparison. For all the models we are using the same optimizer, loss function and the general procedure is the same. For selecting the number of epochs we utilize early stopping. We define a patience factor of 3 meaning that if the loss function is not decreasing for the next 3 consecutives iterations we stop the training and save the model. The best model is considered the one with the lowest loss. We use validation loss to track the overall loss of the model but all the metrics and results are conducted on the test set. For hyper parameter tuning we used as a starting point the weights of the author and then via trial and error we adjusted them to our models.

Model Evaluation

We use the following evaluation metrics for comparing results:

Micro and macro averaged F1 score.

$Micro - R = \frac{\sum_{l=1}^{|L|} TP_l}{\sum_{l=1}^{|L|} TP_l + FN_l}$,

$Macro - R = \frac{1}{|L|} \sum_{l=1}^{|L|} \frac{TP_l}{TP_l + FN_l}$ Where L is the number of ICD codes, TP = True positives and FN=false negatives

We also calculate the following metrics: Macro/Micro Accuracy, Recall, Precision, AUC

Implementation

The data pre-processing was done in Scala using Spark. The extraction of top50 dataset from the preprocessed dataset was done using a combination of python and Scala functions. For model training and evaluation, python was utilized along with pytorch framework both on localhosts and GCP (Google Cloud Platform).

III. Results & Discussion

This section focuses on the results of different architectures tested (logistic regression model, vanilla CNN, RNN, CNN with attention mechanism).

The central intuition for using CNN for text classification is to see hospital notes as images. Let us say we have a sentence and we have maxlen = 70 and embedding size = 300. We can create a matrix of numbers with the shape 70x300 to represent this sentence. For images, we also have a matrix where individual elements are pixel values. Instead of image pixels, the input to the tasks is sentences or documents represented as a matrix. Each row of the matrix corresponds to one-word vector.

CNN works well for Text Classification. It takes care of words in close range. However, it still can't take care of all the context provided in a particular text sequence. It still does not learn the sequential structure of the data, where every word is dependent on the previous word. Or a word in the previous sentence. RNN help us with that. They can remember previous information using hidden states and connect it to the current task.

GRU are a subclass of RNN, specialized in remembering information for an extended period. Moreover, the Bidirectional GRU keeps the contextual information in both directions which is pretty useful in text classification task. Our vanilla CNN model has 100 filters (kernel size=4) as compared to 50 Bi-Directional GRU nodes (kernel size=10) for our vanilla RNN. With this configuration the CNN model out-performs the RNN model marginally, but given the advantages of sequential learning on textual data, tuning the RNN model can help it easily beat the CNN model, but due to lack of experimentation time it has been left to be explored as future work.

The best model selected was CNN with attention very similar to what the author did in his work. The model achieves best scores to almost every metric. We compared results from all our models with the same input, the same loss: Binary Cross Entropy and we achieved the following results as illustrated in the section below. The outcome is very close to the results the author obtained indicating proper pipeline and execution of the steps outlined. Some of our results are better than the authors due to slightly different dataset preprocessing. We added one step of excluding anonymized fields between brackets. We also have less samples in the original data set when filtering from nan values and unique admission ids.

Model	Macro Accuracy	Macro Recall	Macro Precision	Macro F1	Macro AUC	Micro Accuracy	Micro Recall	Micro Precision	Micro F1	Micro AUC
lr	0.288	0.349	0.632	0.450	0.888	0.375	0.446	**0.703**	0.546	0.914
Vanilla CNN	0.463	0.621	0.624	0.622	0.911	0.515	0.684	0.676	0.680	0.935
CNN attn	**0.483**	**0.637**	**0.642**	**0.640**	**0.921**	**0.526**	**0.690**	0.688	**0.689**	**0.941**
Bi-DI GRU	0.390	0.481	0.628	0.545	0.892	0.447	0.555	0.697	0.618	0.919

Table 1: Results on MIMIC 3 top 50 labels

CNN in overall achieves very good results in comparison with logistic regression model and GRU. One reason is the nature of the CNN and the weight sharing. This way the CNN achieves location invariant features where in NLP is translated to words with special meaning independently of their position in the sentence. GRU on the other hand although very computationally demanding did not achieve great results.

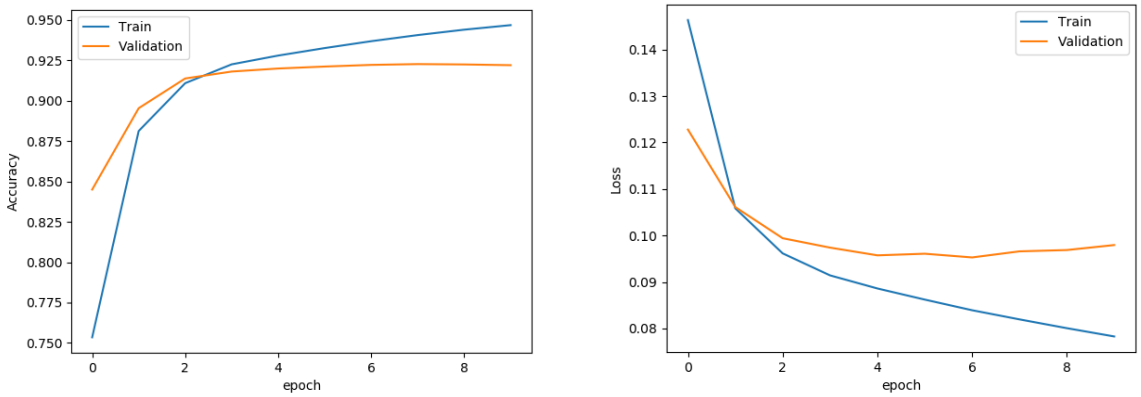

Figure 5: Macro AUC (left) and Binary Cross Entropy loss (right) vs epochs for CNN with attention mechanism

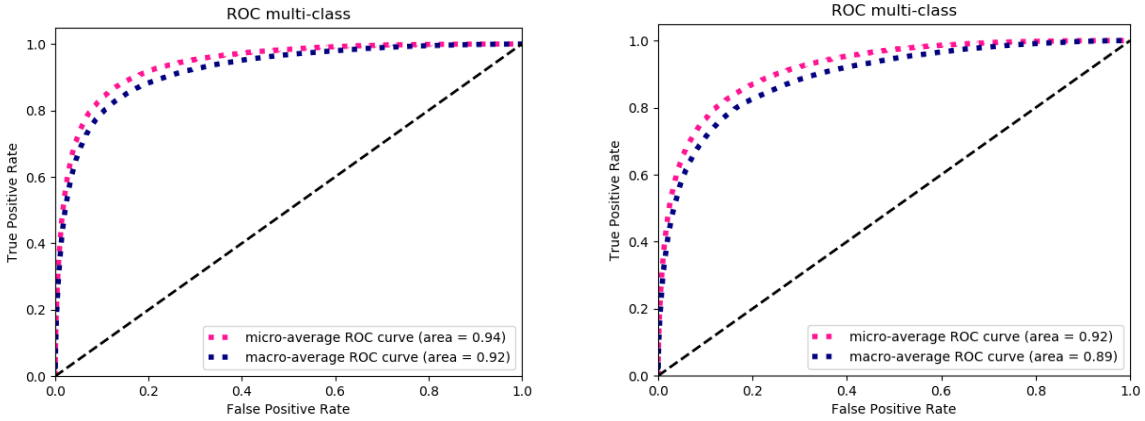

Figure 6: ROC for CNN with attention (left) and ROC for bi-di-GRU (right)

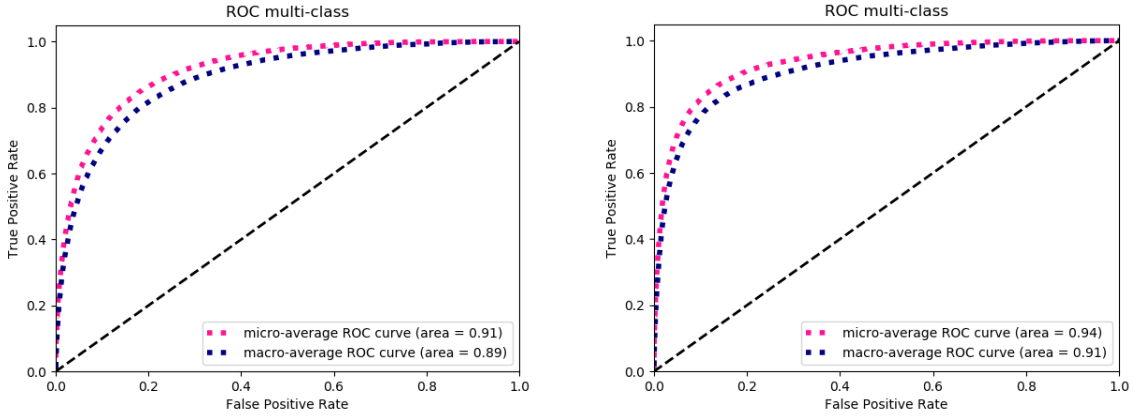

Figure 7: ROC for lr (left) and ROC for vanilla CNN (right)

III. Conclusions

In the specific project a multiclass multi label deep learning model was built to predict the ICD-9 codes from the discharge summaries (clinical notes). We evaluated the tagging of the top 50 codes from the MIMIC 3 dataset on

sequences of different length and with a max length of 2500. Our best model is CNN with attention which achieved macro precision of 0.642 and micro precision of 0.688. CNN with attention mechanism provided the best results. This indicates that the assumption that "*important information is hidden in small snippets of text and for every label they are different*" is correct. Some words have special meaning for a specific label but not for others. In this way we try to find the most important words for every label and then decide if it is tagged with it or not. Further work could be focused on the full dataset of more than 8000 labels. Given the fact the total number of samples is about 53000 and the number of classes is more than 8000, the problem is very difficult. Some anti-measures would be to use the labels descriptions and word2vec to create a similar to logistic regression model to embed the weights to the actual model. By introducing a regularization term in loss function would allow the rarely observed labels to have parameters similar to those of other codes and similar descriptions. Furthermore the use of pretrained weights using models specifically for diagnostic and medication purposes would probably increase the accuracy.

Supporting Information:

Datasets for running training/testing phase of the models: https://gtvault-my.sharepoint.com/:u:/g/personal/astathopoulos3_gatech_edu/ESxAh662fMpAg8zT2gTJhWIBeExI1weYFci0p18rrKTefQ?e=NNfy6F

Github for code: https://github.com/stathopoan/bd4h-cse-6250

Powerpoint presentation: https://gtvault-my.sharepoint.com/:u:/g/personal/astathopoulos3_gatech_edu/ESxAh662fMpAg8zT2gTJhWIBeExI1weYFci0p18rrKTefQ?e=NNfy6F

Youtube Link : https://youtu.be/1aYPVbVvgC0

References

Adler Perotte, R. P. (2013). Diagnosis code assignment: models and evaluation metrics. (pp. 231–237). Journal of the American Medical Informatics Association.

Dzmitry Bahdanau, K. C. (2015). Neural machine translation by jointly learning to align and translate. International Conference on Learning Representations.

Haoran Shi, P. X. (2017). Towards Automated ICD Coding Using Deep Learning.

James Mullenbach, S. W. (2018). Explainable Prediction of Medical Codes from Clinical Text. ACL Anthology.

Kim, Y. (2014). Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1746–1751.

Miltiadis Allamanis, H. P. (2016). A convolutional attention network for extreme summarization of source code. International Conference on Machine Learning, 2091–2100.

T. Kang, S. Z. (2017). Eliie: An open-source information extraction system for clinical trial eligibility criteria. Journal of the American Medical Informatics Association, pp. 1062–1071.

T. Mikolov, K. C. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv.

Tal Baumel, J. N.-K. (2018). Multi-label classification of patient notes a case study on ICD code assignment. AAAI Workshop on Health Intelligence.

Tomas Mikolov, I. S. (2103). Distributed Representations of Words and Phrases and their Compositionality. Advances in neural information processing systems, pp. 3111–3119.

Zichao Yang, D. Y. (2016). Hierarchical Attention Networks for Document Classification. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489.

Team contributions

Tasks

Team member	Task
Anastasios Stathopoulos	• Created the preprocessing pipeline • Worked in training/testing pipeline • Created CNN vanilla model and lr model • Worked in final paper and presentation • Sent mail to get feedback from mentor on next steps
Sagar Jounkani	• Worked in training/testing pipeline • Payed particular attention in evaluating metrics • Created the structure for train/test pipeline • Created CNN attn. and GRU-Bi Di
Ashwin Malgaonkar	• Worked on alternate ways of data preprocessing for possible improvement. • Reviewed the basic preprocessing pipeline. • Worked in Vanilla CNN • Worked on final paper and presentation
Luis Manuel Garcia-Baquero Corredera	• Worked on the implementation of pre-trained word embeddings • Investigated possible causes of bad performance in the draft • Worked in CNN with attention

Individual feedback

Team member	Feedback
Anastasios Stathopoulos	Although the timezones were a lot different and despite the results and the difficulties we faced we managed via communication and task distribution to achieve very good results and finish the project. Special thanks to Sagar Jounkani who have 2 courses on this Semester and still managed to make it happen.
Sagar Jounkani	Thanks to all the team members for taking out time to participate in the project. Anastasios did a very good job in the data pre-processing stage and initializing the modelling process. Ashwin and Luis participated earnestly in the pre-processing, documentation and creation of final presentations. It was really rewarding working with this team.
Ashwin Malgaonkar	Thanks to all the team members in this project. We were able to work through time zones and on different parts of the code. At times we had to work on overlapping tasks and had to compromise in the interest of time to achieve faster comparative results but were able to achieve good results.
Luis Manuel Garcia-Baquero Corredera	After going through different issues we made it to replicate the original paper and achieve a good performance. In my case, it's been very difficult to combine this course with the other 2 courses I had this semester and the FT job, and I think that, because of this, my contribution has not been as relevant as it was supposed to. Thanks to my colleagues for understanding this and making everything advance properly.