Contents

Indian Institute of Technology Jodhpur

Pothula Akash

Assignmnet-2
(M22MA007)

March 7, 2024

Natural Language Processing

1 Detecting Adverse Drug Events using BiLSTM

1.1 colab link :

colab link: Colab link for Assignmnet-2

Figure 1: colab link is provided above

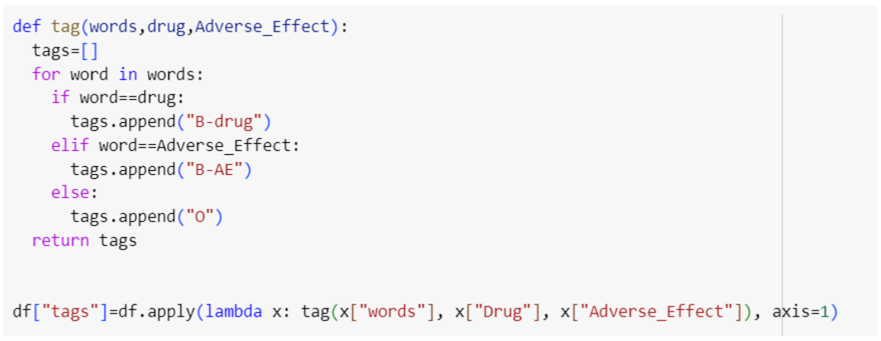
1.2 Preprocessing ADE Dataset :

steps-1: split the sentence into words.

steps-2: Tag corresponding Entities based on the labels provides in adverse effect column and drug column.

Challenges: Exception Entities like "increased calcium-release" are present in the dataset we want to give this whole word a single Enitity tag but if we use simple split function this single word will be divided into multiple words. so we need write our own split function that will handle these kind of words.

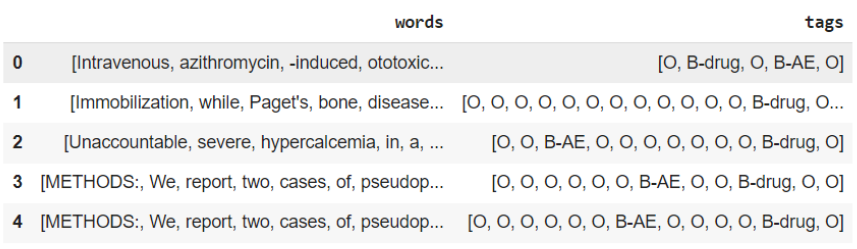
Solution to Challenge: Find entity words in the sentence and wrap the word with brackets. now when you are using split function write exceptions to consider words in the bracket as single word.After splitting is completed just remove the brackets.



```
def tag(words,drug,Adverse_Effect):
  tags=[]
  for word in words:
    if word==drug:
      tags.append("B-drug")
    elif word==Adverse_Effect:
      tags.append("B-AE")
    else:
      tags.append("O")
  return tags


df["tags"]=df.apply(lambda x: tag(x["words"], x["Drug"], x["Adverse_Effect"]), axis=1)
```

Figure 2: B-d,B-AE means drug and adverse effects respectively



	words	tags
0	[Intravenous, azithromycin, -induced, ototoxic...	[O, B-drug, O, B-AE, O]
1	[Immobilization, while, Paget's, bone, disease...	[O, O, O, O, O, O, O, O, O, O, O, O, B-drug, O...
2	[Unaccountable, severe, hypercalcemia, in, a, ...	[O, O, B-AE, O, O, O, O, O, O, O, B-drug, O]
3	[METHODS:, We, report, two, cases, of, pseudop...	[O, O, O, O, O, O, B-AE, O, O, B-drug, O, O]
4	[METHODS:, We, report, two, cases, of, pseudop...	[O, O, O, O, O, O, B-AE, O, O, O, O, B-drug, O]

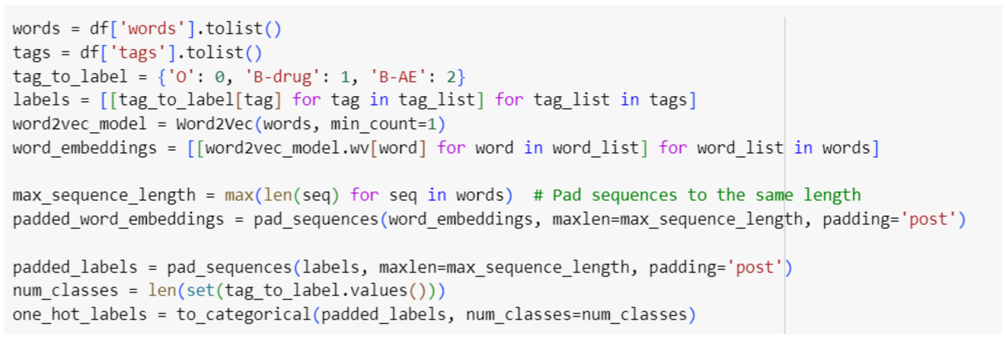
Figure 3: Final Tagging

1.3 Word2Vec and one-hot encoding:

steps-1: Convert all the tokens obtained from the sentence into vector for using gensim Library.

steps-2: Convert all the enities tagged into one-hot encoding (we have 3 labels).

step-3: Pad all the vectors and labels.



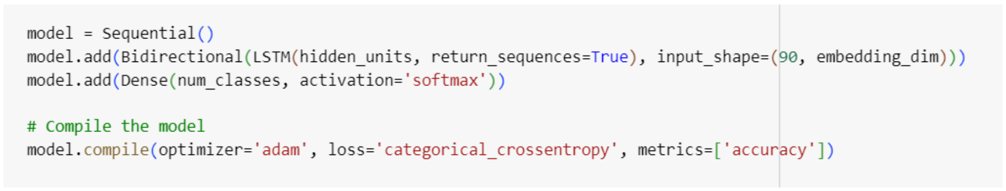
```
words = df['words'].tolist()
tags = df['tags'].tolist()
tag_to_label = {'O': 0, 'B-drug': 1, 'B-AE': 2}
labels = [[tag_to_label[tag] for tag in tag_list] for tag_list in tags]
word2vec_model = Word2Vec(words, min_count=1)
word_embeddings = [[word2vec_model.wv[word] for word in word_list] for word_list in words]

max_sequence_length = max(len(seq) for seq in words)  # Pad sequences to the same length
padded_word_embeddings = pad_sequences(word_embeddings, maxlen=max_sequence_length, padding='post')

padded_labels = pad_sequences(labels, maxlen=max_sequence_length, padding='post')
num_classes = len(set(tag_to_label.values()))
one_hot_labels = to_categorical(padded_labels, num_classes=num_classes)
```

Figure 4: Converted to Embedding

step-4: Build a BiLSTM model.



```
model = Sequential()
model.add(Bidirectional(LSTM(hidden_units, return_sequences=True), input_shape=(90, embedding_dim)))
model.add(Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 5: BiLSTM mdoel

1.4 Evaluation:

Accuracy: Average Accuracy across all Outputs is 98.48%

Precision: Average Precision across all Outputs is 98.26%

Recall: Average Recall across all Outputs is 98.48%

F1-score: Average F1-score across all Outputs is 98.27%

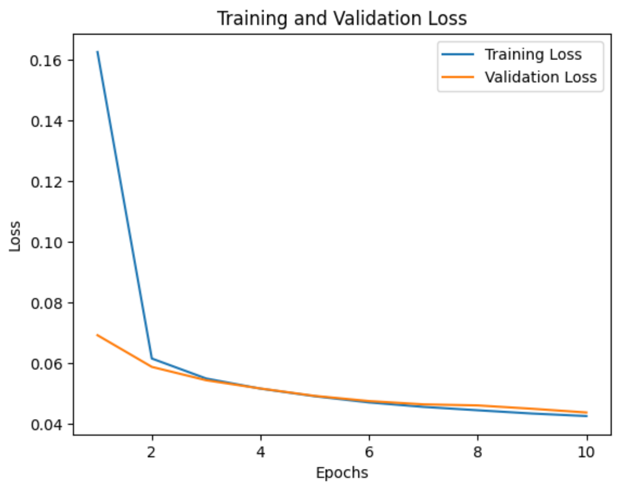


Figure 6: Train and Val Loss Curve

Receiver Operating Characteristic Curve:

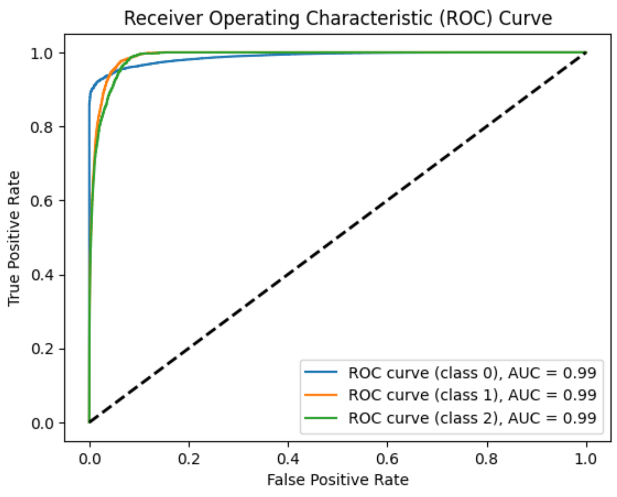


Figure 7: ROC Curve

1.5 Comparing Glove with Word2Vec:

Accuracy: Average Accuracy across all Outputs using Glove is 99.2%

Precision: Average Precision across all Outputs using Glove is 99.26%

Recall: Average Recall across all Outputs using Glove is 99.48%

F1-score: Average F1-score across all Outputs using Glove is 99.27%

How to improve?

we can finetune these pre-trained models in our own domain data to improve the performance of model.

Working Comparison:

Glove is based on co-occurrence of the words in the corpus.

Gensim's Word2vec uses method like Continuous bag of words and skip gram to get the word embeddings.

Training time Comparison:

Glove requires more time to contruct the co-occurrence matrix.

Gensim's Word2Vec requires less time as it is only simple feed forward neural network.

Pretrained model Comparison:

Glove pre-trained models is trained on multiple languages and multiple domains so it is used very widely

Gensim's Word2Vec less used trained on less domains.

2 Literature Review:

2.1 PAPER-1: Adverse drug event detection using natural language processing: A scoping review of supervised learning methods:

key Information:

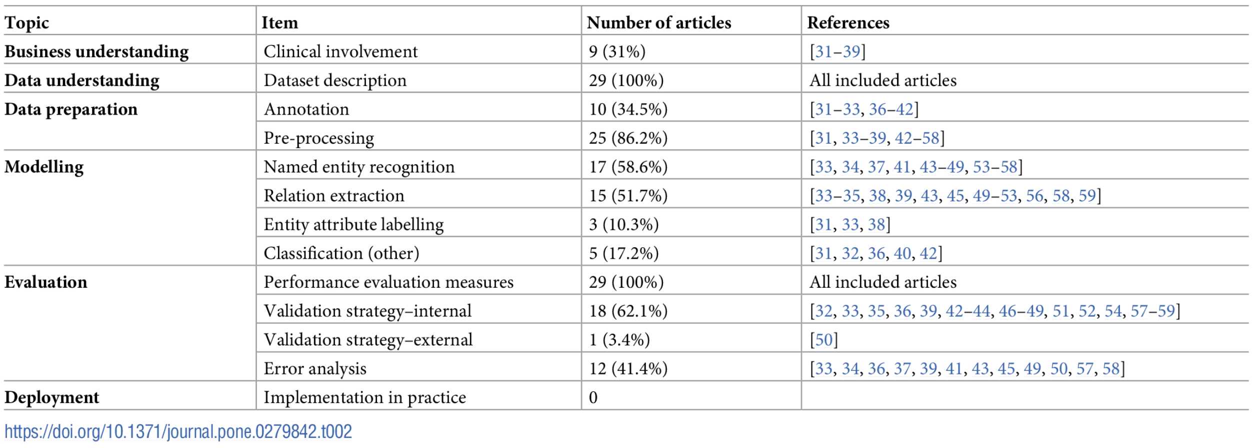
1) This review tells about how natural language processing can be used for detecting Adverse Drug Events in hospital recods in early stages.

2)Annotation is very hectic task. so this review suggested to buid a semi-automatic nlp model to annotate the data.

3) For detecting ADE nlp is using LSTM and CRF methods but the main problem it is only working good for english language and not for other languages.

4) Researchers encountering difficulties with using pre-processing tools in clinical text analysis should build their own custom nlp tools tailored to the specific domain and language.

5) To improve accuracy of annotation models for ADEs we should consider treating ADEs as relations between a drug(azithromycin) and symptom entity(ototoxicity) , rather than as seperate entities.This approach will align with how doctors interpret clinical data and reduces errors associated with ambiguous labeling of symptoms as ADEs.



Topic	Item	Number of articles	References
Business understanding	Clinical involvement	9 (31%)	[31–39]
Data understanding	Dataset description	29 (100%)	All included articles
Data preparation	Annotation	10 (34.5%)	[31–33, 36–42]
	Pre-processing	25 (86.2%)	[31, 33–39, 42–58]
Modelling	Named entity recognition	17 (58.6%)	[33, 34, 37, 41, 43–49, 53–58]
	Relation extraction	15 (51.7%)	[33–35, 38, 39, 43, 45, 49–53, 56, 58, 59]
	Entity attribute labelling	3 (10.3%)	[31, 33, 38]
	Classification (other)	5 (17.2%)	[31, 32, 36, 40, 42]
Evaluation	Performance evaluation measures	29 (100%)	All included articles
	Validation strategy–internal	18 (62.1%)	[32, 33, 35, 36, 39, 42–44, 46–49, 51, 52, 54, 57–59]
	Validation strategy–external	1 (3.4%)	[50]
	Error analysis	12 (41.4%)	[33, 34, 36, 37, 39, 41, 43, 45, 49, 50, 57, 58]
Deployment	Implementation in practice	0	

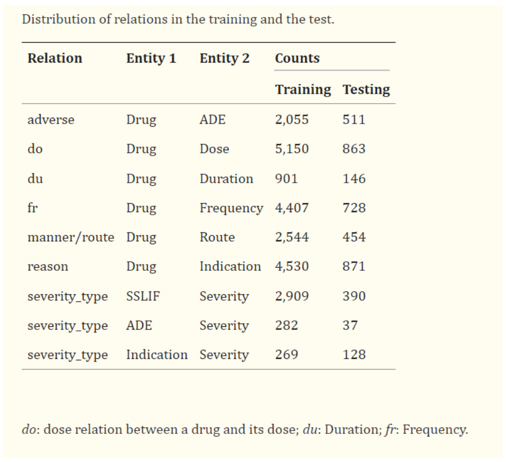
https://doi.org/10.1371/journal.pone.0279842.t002

Figure 8: Summary of main findings

2.2 PAPER-2: A System for Detecting Medications, Adverse Drug Events, and their Relations from Clinical Notes

Dataset details:

1) Dataset is provided with a details like drug and corresponding adverse effect, drug and dosage info, drug and frequency of drug administration.



Distribution of relations in the training and the test.

Relation	Entity 1	Entity 2	Counts	
			Training	Testing
adverse	Drug	ADE	2,055	511
do	Drug	Dose	5,150	863
du	Drug	Duration	901	146
fr	Drug	Frequency	4,407	728
manner/route	Drug	Route	2,544	454
reason	Drug	Indication	4,530	871
severity_type	SSLIF	Severity	2,909	390
severity_type	ADE	Severity	282	37
severity_type	Indication	Severity	269	128

do: dose relation between a drug and its dose; *du*: Duration; *fr*: Frequency.

Figure 9: dataset details

Workflow of NER module

1) First we detect the sentence boundary and then tokenise the sentence. Tokensizing will cause a problem.

2) suppose you want to identify a drug ”100mg of ibuprofen” but text is tokensied and divided this single drung name into 3 tokens so to avoid this a positional mapping file is used which helps in identifying such words as a single entity. 3) now apply model as shown in fig below

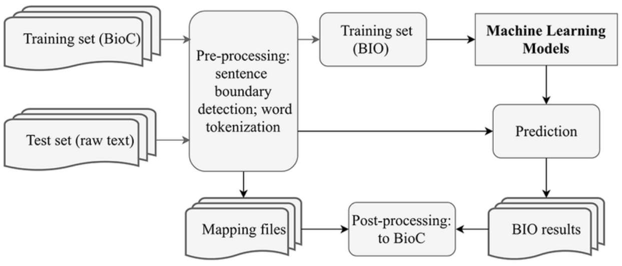


Figure 10: workflow of NER module

Relation Extraction Module:

1) Main aim of the Relation Extraction Module, suppose a ”patient” was prescribed with a medicine ”ibuprofen” this module will identify relation called ”prescribe” between two entities patient and ibuprofen.

2) Relation Extraction Module contains 3 steps a pre-processing pipeline, a classifier, and a post-processing pipeline.

3) pre-processing step involves Heuristic Rules

Heuristic Rules to Generate Concept Pairs :

when you are creating pairs with all the data you will get lot of pairs like (patient,is), (patient,was), (patient,ibuprofen) but most of them are unrelated only (patient,ibuprofen) realtion is what we want in order to avoid getting unrelated pairs we use heuristic.

4) Two clinical concepts occurred in the same sentence or two consecutive sentences will be considered as a candidate pair and continue to Rule 2; otherwise, stop;

5) For each of the pairs generated in Rule 1, if the entity types of the two concepts fall into any possible combinations shown in fig 9, it will be a candidate pair for classification; otherwise, stop;

6) After pre-processing next step is classifying , used SVM and RF to extract features like the local context information including words inside of each entity, the distance between two clinical concepts in number of characters and in number of words, unigram, bigram, and trigram before and after each entity, semantic information.

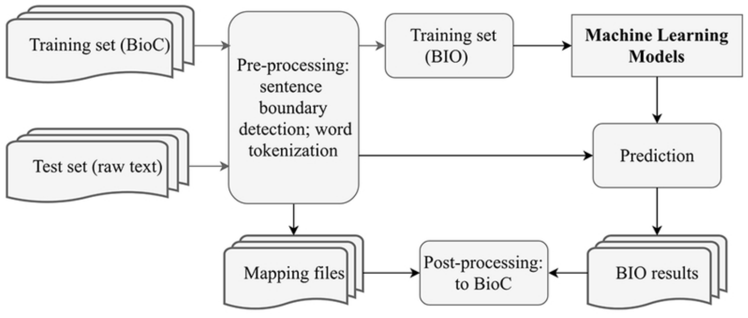


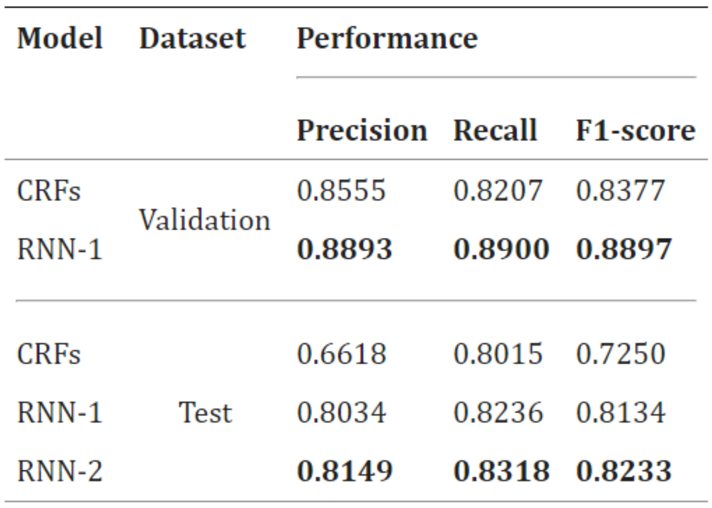
Figure 11: workflow of Relation Extraction module

Integrated pipeline:

Integrate relation extraction with NER module and unify them to extract clinical concetpts along with their relation from clinical text. In integrated module , the relation extrction is based on clinical concepts detected by the NER modeule.

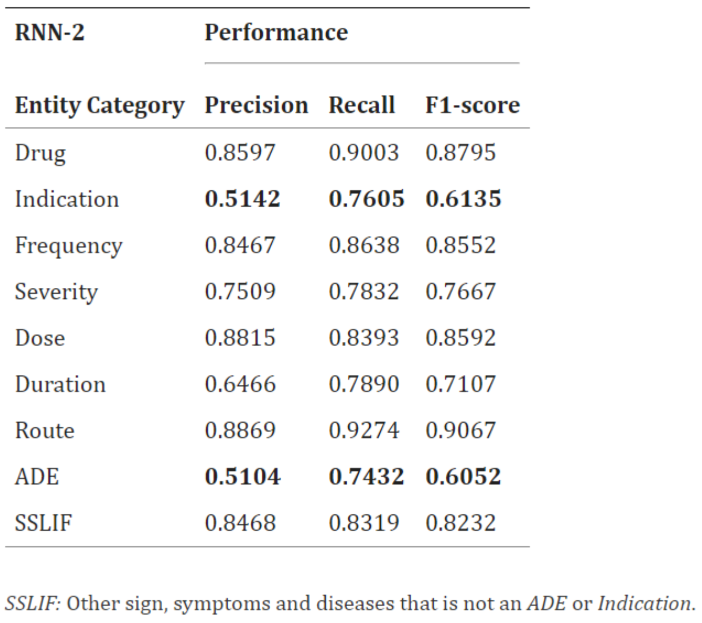
Evaluation:

Both RNN-1 and RNN-2 outperformed the baseline CRFs model by about 0.1 in terms of strict F1-score.



Model	Dataset	Performance		
		Precision	Recall	F1-score
CRFs	Validation	0.8555	0.8207	0.8377
RNN-1		**0.8893**	**0.8900**	**0.8897**
CRFs	Test	0.6618	0.8015	0.7250
RNN-1		0.8034	0.8236	0.8134
RNN-2		**0.8149**	**0.8318**	**0.8233**

Figure 12: The NER performances on the validation and test.



RNN-2	Performance		
Entity Category	Precision	Recall	F1-score
Drug	0.8597	0.9003	0.8795
Indication	**0.5142**	**0.7605**	**0.6135**
Frequency	0.8467	0.8638	0.8552
Severity	0.7509	0.7832	0.7667
Dose	0.8815	0.8393	0.8592
Duration	0.6466	0.7890	0.7107
Route	0.8869	0.9274	0.9067
ADE	**0.5104**	**0.7432**	**0.6052**
SSLIF	0.8468	0.8319	0.8232

SSLIF: Other sign, symptoms and diseases that is not an *ADE* or *Indication*.

Figure 13: Performances of RNN-2 on the test set for each entity type

2.3 PAPER-3: Bidirectional LSTM-CRF for Adverse Drug Event Tagging in Electronic Health Records

Aim:

Aim of the paper is to build a NER system that is capable of detecting any medication names and their attributes as well as ADEs.

Dataset details:

1) Dataset has Total of 1089 number of EHR notes took from 21 cancer patients

The EHR notes are annotated with the following information: Medication information: Including medication name, dosage, route, frequency, and duration ,Adverse Drug Events (ADEs), Indications, Other signs and symptoms

2)The major challenges with processing the EHR records is to extracting entities from such narratives. This notes often contain medical and non-medical abbreviations, acronyms and misspelled words which will cause difficulty in extracting entities.

Ambiguity is also a problem as shown below :

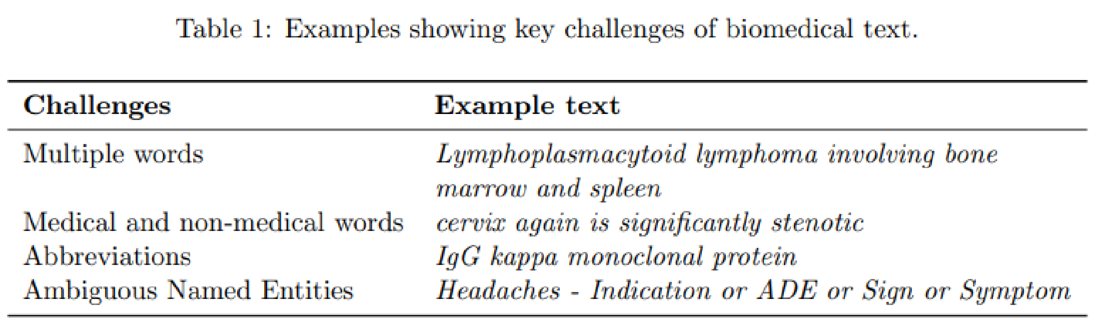


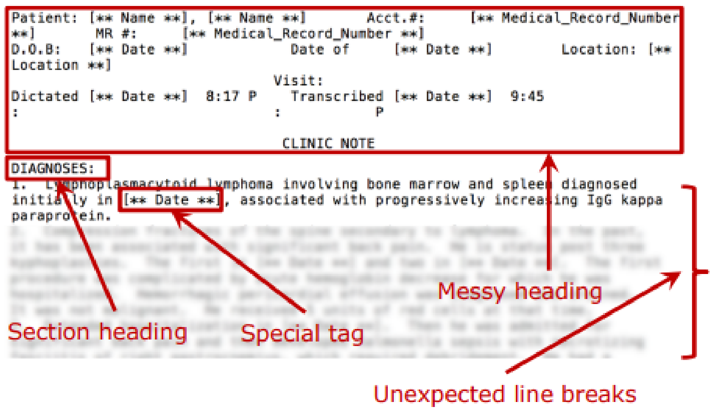
Table 1: Examples showing key challenges of biomedical text.

Challenges	**Example text**
Multiple words	*Lymphoplasmacytoid lymphoma involving bone marrow and spleen*
Medical and non-medical words	*cervix again is significantly stenotic*
Abbreviations	*IgG kappa monoclonal protein*
Ambiguous Named Entities	*Headaches - Indication or ADE or Sign or Symptom*

Figure 14: dataset details

Dataset Preprocessing:

1) Convert EHR notes to sentences and then to words. preprocessing involves dealing with repeated punctuations around section headings , unexpected line breaking, bad text formatting in this cases NLTK will fail very badly but this model dealt with all these issues.



```
Patient: [** Name **], [** Name **]        Acct.#:      [** Medical_Record_Number
**]        MR #:      [** Medical_Record_Number **]
D.O.B:   [** Date **]              Date of      [** Date **]          Location: [**
Location **]
                                  Visit:
Dictated [** Date **]  8:17 P     Transcribed [** Date **]  9:45
:                                 :              P

                                   CLINIC NOTE
DIAGNOSES:
1.  Lymphoplasmacytoid lymphoma involving bone marrow and spleen diagnosed
initially in [** Date **], associated with progressively increasing IgG kappa
paraprotein.
[illegible]
```

Messy heading
Section heading
Special tag
Unexpected line breaks

Figure 1: Noise in the EHR text.

Figure 15: Noise in the EHR text

Model and Methodology:

1) model is composed of a bi-LSTM neural network for an input layer responsible for character embedding, a second bi-LSTM for word embedding followed by a linear-chain CRF output layer.

2) convert each word to embedding feed this embedding into BiLSTM. then this will extract meaning and contextual information from each word in the sentence. The output from BiLSTM will be fed into feed forward neural network to get probability distributions, where each entry will correspond to the a score with the tag (tags are entities) then this output will be fed into linear-chain CRF. now train the model by minimising the loss.

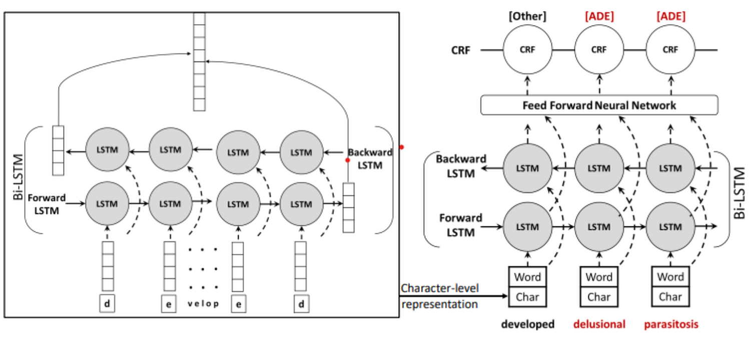


Figure 16: DLADE System Architecture

Evaluation:

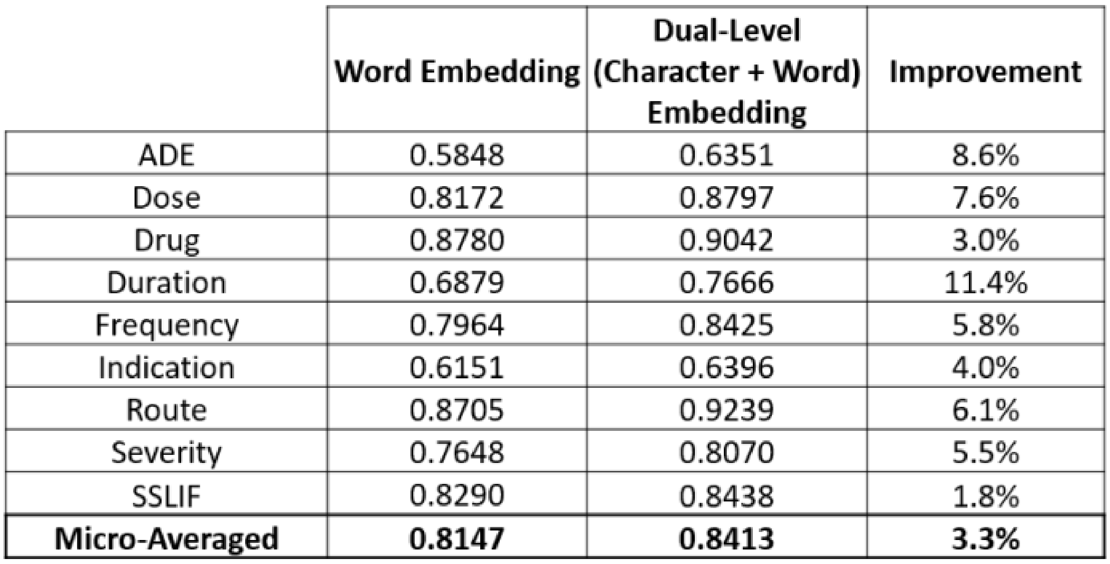
DLADE system micro-averaged precision of 0.8373

DLADE system micro-averaged recall of 0.8373

DLADE system micro-averaged F1-score of 0.8413

performance surge in DLADE's is due to utilizing both learned character-level representations and pre-trained word-level embeddings.

paper demonstrates the effectiveness of integrating two complementary sequence labeling techniques, combined with dual-level embeddings(both character level and word level), for accurately extracting information from Electronic Health Record (EHR) notes.



	Word Embedding	Dual-Level (Character + Word) Embedding	Improvement
ADE	0.5848	0.6351	8.6%
Dose	0.8172	0.8797	7.6%
Drug	0.8780	0.9042	3.0%
Duration	0.6879	0.7666	11.4%
Frequency	0.7964	0.8425	5.8%
Indication	0.6151	0.6396	4.0%
Route	0.8705	0.9239	6.1%
Severity	0.7648	0.8070	5.5%
SSLIF	0.8290	0.8438	1.8%
Micro-Averaged	**0.8147**	**0.8413**	**3.3%**

Figure 17: Improvement for MADE 1.0 in F1-score when using Dual-Level Embeddings.

References