Medical Entity Recognition in Tweets

[Project Deliverable 2]

Team #19 Members:

1. Ankit Misra (2018201079)
2. Harshita Agarwal (2018201014)
3. Souparna Das (2018201010)
4. Supriya Priyadarshani (2018202009)

Mentor: Nikhil Pattisapu

github: https://github.com/ankitmishra16/Medical-Entity-Recognition

ABSTRACT

Medical Entity Recognition is a crucial step towards efficient medical texts analysis. For this project, the medical entities which might be one single term or a sequence of terms should be one of the following categories:

- Disease
- Symptoms
- Drugs
- Test
- Treatment

Social media sites, such as Twitter, can be a rich source of many kinds of information, including health-related information. Accurate detection of entities such as diseases, drugs, and symptoms could be used for biosurveillance (e.g. monitoring of flu) and identification of adverse drug events. This task becomes difficult on tweets as they are known to be full of slang and doesn't guarantee enough context for information extraction.

PROBLEM STATEMENT

Named Entity Recognition can automatically scan entire articles and reveal which are the major people, organizations, and places discussed in them. Knowing the relevant tags for each article help in automatically categorizing the articles in defined hierarchies and enable smooth content discovery. Medical Entity Recognition (MER) aims to identify and classify the portions of text which span medical entity mentions. The entity types are fixed and known in advance. In this project the entity types to be recognized are Drug, Disease, Symptom, Treatment and Test. Traditional MERs are used on formal medical texts, such as published medical studies, research articles, clinical reports, discharge summaries and Electronic Health Records. In this project, we will evaluate the performance of various existing MERs on Twitter data and try to improve its performance on tweets.

DATASETS

- **CADEC dataset :**

This dataset has 1250 files of text and its annotations. In CADEC dataset we were given data in form of 'brat-flavored standoff', with .txt and .ann extension files, we have to parse it to convert it to usable (CSV) format. For this part we had used this script. It reduces the 'brat-flavored standoff' to a text format with labelling of every word, using the .ann file. Then we have to again parse it to convert it to CSV format. For this we had written a script named 'data_processing.ipyb', and we also have to handle few errors in it, as this script makes multiple entries for the same data, so we have to handle this error specifically.

- **Micromed dataset :**

This dataset is of .linejson format, with tweet IDs as keys, and 'type' contains type of annotation that a word belongs to, and an attribute of json type named 'location' has two keys 'start' and 'end', which tells us where in the text that word is starting and where it is ending. But in this field, data has a lot of discrepancies, as the pointers are very different from the actual occurrences of the words, for this we had to go manually, file by file where it occurs, to resolve it. And after doing all this we got only 370 tweets, as many of them were either deleted or the account from which they were tweeted was deactivated.

After doing all this, again we had converted it to 'brat-flavored standoff' format, as we had a ready script to get CSV from 'brat-flavored standoff' format, and after doing all this preprocessing we got ready to use CSV format of Twitter data.

- **Twimed dataset :**

This is another twitter dataset which we got from this link, it is again 'brat-flavored standoff' formated .ann file, with named on their tweet IDs, so using their names we had fetched the tweets(only 521 were present), and created a .txt file for each tweet so that we can use our previous script. After that, we had just used our previous script to convert it to CSV. This data also had a discrepancy but it was only one, so it doesn't make much trouble to us.

BASIC SOLUTION APPROACHES

For the given problem statement, we need to recognize the terms which is related to the medical fields. For the project purpose, we will be using 5 categories of

medical entities. The 5 categories are: **1. Drugs 2. Symptoms 3. Treatments 4. Tests 5. Disease**. So, each medical entity will be labeled as one of these 5 categories.

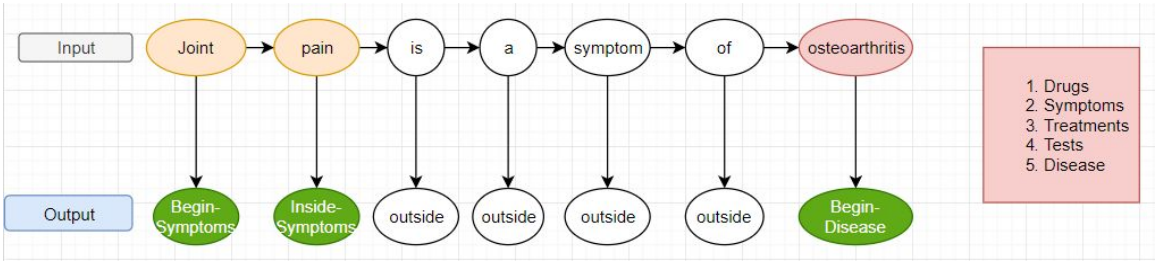
The terminologies which are used to recognize medical entities are as follows:

1. **Begin**: signifies that this term is the beginning of a medical entity
2. **Inside**: This signifies that the current term is inside a medical entity or a part of medical entity.
3. **Outside**: This signifies that a term is not a part of any medical entity.

Let us consider the following statement:

"Joint pain is a symptom of osteoarthritis"

If we provide the above as an input, we want our solution as follows:



Thus, the problem can be easily be reduced to a sequence problem. Which can be solved using neural networks like LSTM or standard machine learning generative models like Hidden Markov Models. Once the basic structure is done, to enhance the performance, various techniques can be incorporated. For example, if a large list of medical terms are already provided, then it can be used to increase the performance.

The final deliverable will be based on Twitter data, which can be very noisy and skewed. The idea is to filter out relevant tweets and adjust model's hyperparameters to obtain desired results. Another method is to identify common suffixes or prefixes for a group of medical terms. For example, "Arithromycyn ", "Paracetamol", "zerodol" have common suffixes or prefixes with other drugs name. Even the slightest change can be critical for performance enhancement.

METHODOLOGIES USED (Different from the methods specified in the initial scope document)

In the initial scope document , we had planned to use LSTM and HMM models, but we experimented and came up with other hybrid models too, that performed comparatively better than the simple LSTM model. Below are the new models :

We have used two methodologies so far.

1. Bi-LSTM Model

Another important strategy in building a high-performing deep learning method is understanding which type of neural network works best to tackle NER problem considering that the text is a sequential data format. yeah, you guessed it right... Long short Term Memory (LSTM). more details about LSTMs in this link. But not any type of LSTM, we need to use bi-directional LSTMs because using a standard LSTM to make predictions will only take the "past" information in a sequence of the text into account. for NER, since the context covers past and future labels in a sequence, we need to take both the past and the future information into account. A bidirectional LSTM is a combination of two LSTMs — one runs forward from "right to left" and one runs backward from "left to right". So, the whole context information can be captured using bi-directional LSTM unlike the traditional LSTM model.

The model which have been used as the first step is as follows:

- 40 dimensional word-embeddings have been used.
- The max length of each sentence is restricted to 75 words.
- 128 nodes of bi-directional LSTM is used.
- A dense fully connected layer.

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_10 (Embedding)     (None, 75, 40)            266600
_________________________________________________________________
```

```
bidirectional_10 (Bidirectio (None, 75, 256)           173056
_________________________________________________________________
time_distributed_10 (TimeDis (None, 75, 11)            2827
=================================================================
Total params: 442,483
Trainable params: 442,483
Non-trainable params: 0
```

OUTPUT RESULTS using the above model:

The result of the model trained and tested on CADEC dataset is as follows:

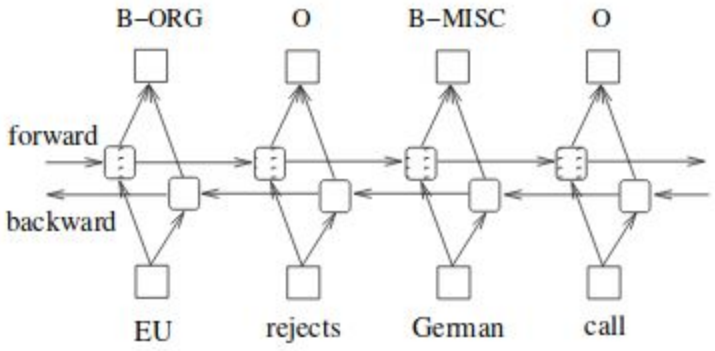
	precision	recall	f1-score	support
B-ADR	0.65	0.68	0.66	594
B-Disease	0.00	0.00	0.00	20
B-Drug	0.96	0.84	0.90	160
B-Finding	0.00	0.00	0.00	49
B-Symptom	0.00	0.00	0.00	23
I-ADR	0.64	0.61	0.63	1085
I-Disease	0.00	0.00	0.00	14
I-Finding	0.00	0.00	0.00	54
I-Symptom	0.00	0.00	0.00	25
O	0.94	0.97	0.96	9663
PAD	1.00	1.00	1.00	44713
accuracy			0.98	56400
macro avg	0.38	0.37	0.38	56400
weighted avg	0.98	0.98	0.98	56400

It is notable that, in cases where some classes are really less in number, the model is still unable to capture such information where the samples are really less. The weighted avg indicates this fact very well.

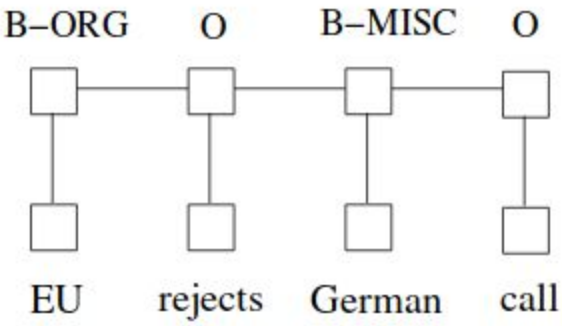
z

2. **Bi-LSTM-CRF Model:**

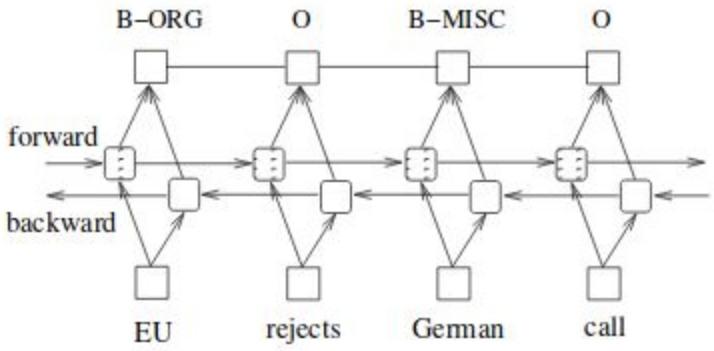
Bidirectional LSTM Networks: In order to feed the text into our Bi-LSTM-CRF, all texts should be the same length. We used the sequence.pad_sequences() method and MAX_LEN variable. All texts longer than MAX_LEN are truncated and shorter texts are padded to get them to the same length.
In sequence tagging task, we have access to both past and future input features for a given time,we can thus utilize a **bidirectional LSTM network.** In doing so, we can efficiently make use of past features (via forward states) and future features (via backward states) for a specific time frame. We train bidirectional LSTM networks using back-propagation through time. The forward and backward passes over the unfolded network over time are carried out in a similar way to regular network forward and backward passes, except that we need to unfold the hidden states for all time steps. We also need a special treatment at the beginning and the end of the data points. In our implementation, we do forward and backward for whole sentences. We have batch implementation which enables multiple sentences to be processed at the same time.



CRF Networks: There are two different ways to make use of neighbor tag information in predicting the current tags. The first is to predict a distribution of tags for each timestep and then use the beam-like decoding to find the optimal tag sequences. The second one is to focus on sentence level instead of individual positions, thus leading to Conditional Random Fields(CRF) models. CRFs can produce higher tagging accuracy in general.



BI-LSTM-CRF Network : **We combined the bidirectional LSTM network and a CRF network to form a BI-LSTM-CRF network** . In addition to the past input features and sentence level tag information that is used in a LSTM-CRF model, a BI-LSTM-CRF model can use the future input features also. The extra features can boost tagging accuracy as we will show in experiments below.



Parameters :

BATCH_SIZE = 512 (Number of examples used in each iteration)
EPOCHS = 100 (Number of passes through entire dataset)
MAX_LEN = 60 (Max length of review (in words))
EMBEDDING = 200 (Dimension of word embedding vector)

- On training the data for CADEC dataset, which is a rich annotated corpus of medical forum posts on patient reported Adverse Drug Events (ADEs) using the above model, and testing it on Twitter dataset gave the following result. The first reason of not so good result is :
    1. Data used for training is small, so the model was not trained up to the mark.
    2. The tags that are used in the training dataset are little different from the Twitter dataset, for ex. ADR and Finding tags are not there in the Twitter dataset, so the precision for them is zero.

Accuracy : 0.8793972939729398 %

	precision	recall	f1-score	support
B-ADR	0.00	0.00	0.00	0
B-Disease	0.00	0.00	0.00	52
B-Drug	0.03	0.02	0.02	309
B-Finding	0.00	0.00	0.00	0
B-Symptom	0.25	0.00	0.01	231
I-ADR	0.00	0.00	0.00	0
I-Disease	0.00	0.00	0.00	38
I-Drug	0.00	0.00	0.00	32
I-Finding	0.00	0.00	0.00	0
I-Symptom	0.15	0.01	0.02	190
O	0.87	0.80	0.83	5582
PAD	1.00	1.00	1.00	9826
accuracy			0.88	16260
macro avg	0.19	0.15	0.16	16260
weighted avg	0.91	0.88	0.89	16260

To train the model better for the Twitter dataset, we need more data for Twitter. Currently, we only have 521 sentences. Even if we use a part of it to add in he CADEC dataset, to train the model better for the Twitter dataset, the model still performs bad.

- On training and testing on the CADEC dataset itself, comparatively better results were obtained. The number of sentences considered is 7520 which still is not too big.

  This result is with maximum length 60 for all the sentences.

Accuracy : 0.9714539007092199 %

	precision	recall	f1-score	support
B-ADR	0.67	0.67	0.67	592
B-Disease	0.37	0.33	0.35	33
B-Drug	0.96	0.85	0.90	170
B-Finding	0.31	0.22	0.26	46
B-Symptom	0.35	0.39	0.37	23
I-ADR	0.63	0.57	0.60	1094
I-Disease	0.23	0.14	0.17	22
I-Drug	0.52	0.33	0.41	33
I-Finding	0.22	0.14	0.17	50
I-Symptom	0.36	0.13	0.19	38
O	0.94	0.96	0.95	10279
PAD	1.00	1.00	1.00	32740
accuracy			0.97	45120
macro avg	0.55	0.48	0.50	45120
weighted avg	0.97	0.97	0.97	45120

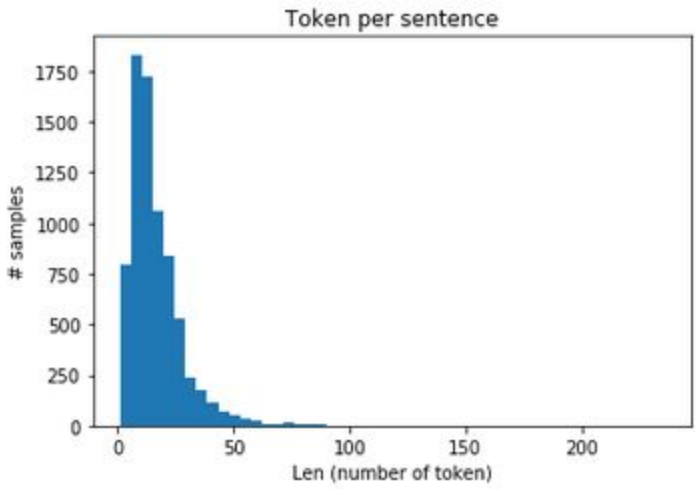
The following result is with maximum length 20 for all the sentences :

Accuracy : 0.9225398936170213 %

	precision	recall	f1-score	support
B-ADR	0.69	0.61	0.65	486
B-Disease	0.45	0.21	0.29	24
B-Drug	0.96	0.83	0.89	132
B-Finding	0.20	0.18	0.19	28
B-Symptom	0.19	0.25	0.22	20
I-ADR	0.59	0.52	0.55	962
I-Disease	0.00	0.00	0.00	13
I-Drug	0.67	0.29	0.40	14
I-Finding	0.26	0.13	0.17	53
I-Symptom	0.15	0.18	0.16	34
O	0.93	0.96	0.94	8076
PAD	1.00	1.00	1.00	5198
accuracy			0.92	15040
macro avg	0.51	0.43	0.46	15040
weighted avg	0.92	0.92	0.92	15040

This result makes sense, because it seems that the major difference in the 2 results is because of the padding (Accuracy in the above result is 97%, and Accuracy with

less maximum length is 92%). But, if carefully observed, the other important tag frequency is also reduced because of the small maximum length considered. So, the maximum length should not be reduced this much just to avoid more padding. The average sentence length for the sample can be seen below in the graph:



FINDINGS

- Till now, the word-embeddings which are used are W2V and keras embedding. The Word2Vec was trained on the CADEC dataset and 300 dimensional word vectors were used. Using Word2Vec has not produced any significant results, one main reason for that is the corpus which has been used, is really small in size. We haven't yet tried with pre-trained W2V with large corpuses. Which we look forward to. Right now, the results are from the keras-embedding. Next we will use more sophisticated embeddings like Gloves, Elmo with pre-trained weights and check the performances.

- We can compare the results of the 2 models for the precision, recall and F1 score per tag. We can observe, that Bi-Lstm-Crf model, predicted all the tags far better than the simple Bi-Lstm model, with an accuracy of 97%. The accuracy in the Bi-Lstm model also seems to be good, but on observing per tag precision, only 'Drug', 'O' and 'PAD' tags are contributing to the accuracy, which is not the case with the Bi-Lstm-Crf model. The BiLSTM-CRF model outperforms the simple BiLSTM models as it can produce results for those classes which have significantly less number of samples.

- Another important aspect is the size of the data. Normally, neural networks often produce biased results if not enough data is fed. The datasets we used are small in size and more and more data will improve the performance for sure. For the twitter dataset, as it is more noisy, more data is required for the model to enhance the performances.

NEW MILESTONES FOR FINAL DELIVERABLE GOALS

- After getting sufficient Twitter data, we will manually(using script), annotate that data using our list of medical entities, and then we will use that data to train our model and test on Twitter data.
- Trying these models for bigger datasets and then testing on Twitter Data.
- Implementing BiLSTM-CNN, BI-LSTM-CNN-CRF model and compare the results with the currently implemented models.
- Evaluation of performance of all models on twitter data.
- Work on possible enhancements for twitter data.

REFERENCES

1. Bidirectional LSTM-CRF Models for Sequence Tagging. https://arxiv.org/pdf/1508.01991v1.pdf
2. Named Entity Recognition with Bidirectional LSTM-CNNs. https://arxiv.org/pdf/1511.08308.pdf
3. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. https://arxiv.org/pdf/1603.01354.pdf
4. A Study of Neural Word Embeddings for Named Entity Recognition in Clinical Text. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4765694/pdf/2248738.pdf
5. Script to convert 'brat-flavored standoff' format to .txt format https://github.com/spyysalo/standoff2conll/blob/master/standoff2conll.py
6. Script to Twimed dataset https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5438461/