Automatic extraction and normalization of adverse effects in English tweets

By: Charlene Hsiao, Jonathan Martindale, Tabassum Nisha, Jingcheng Xiao

INTRODUCTION

Overview:
It is an end-to-end task that involves extracting the span of a text containing an adverse effect of medication from tweets that are report as ADRs, and then mapping the extracted ADRs to a standard concept ID in the MedDRA vocabulary (a clinically validated international medical terminology dictionary).

Data:

- *Training data- includes tweets that report an adverse drug reaction*
- *Validation data- includes tweets along with additional columns such as predicted labels and tokenized cleaned tweets*

Labels/Tags:
Labels/tags act as keys to the respective tokenized tweets. In this task the tag ‘B’ represents the first word of the adverse drug reaction label in the tweet, ‘I’ represents the rest of the words in the adverse drug reaction label and ‘O’ represents all the other words in the tweet.

PART 1 - Training data: Cleaning and Sequence Labeling

1. Cleaning the raw tweet text data by :
    a. Changing all the text to lowercase
    b. Removing @ mentions
    c. Removing all other non-standard characters
    d. Removing all the urls
    e. Expanding contractions

2. Cleaned twitter texts tokenization and lemmatization:
    a. Comparing the performance of the different tokenizers from NLTK package
    b. Choosing not to use the lemmatization technique because of the mis-matching issue

3. Labeling the tokenized twitter with labels “B”,”I”,”O” ,generating the formatted data with tokenized twitters and the matched labels for NER model.

PART 2 - Named Entity Recognition (NER)

Named entity recognition (NER) is the first step towards information extraction that seeks to locate and classify named entities in text into predefined categories, here mentioned as ADRs.

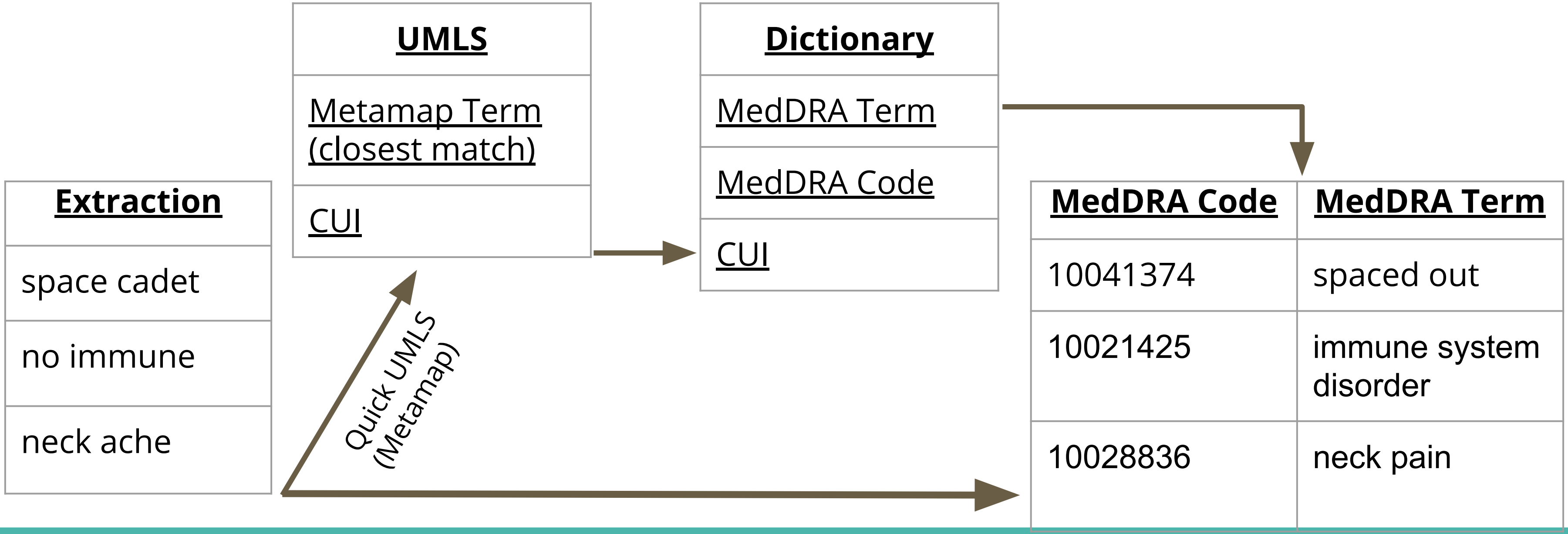
We used the following tools:

1. NER code blocks using flair, pytorch provided by Xinyan Zhao
   a. Flair is a NLP library
   b. Word embedding with GloVE
2. Used NER code to predict the tags “B”, “I”, “O”
3. Tuning the model by varying the hyperparameters such as:
   a. Training_epoch (number of training iterations)
   b. WordEmbeddings
   c. Use_crf (Conditional Random Fields)
   d. Lr (Information Retrieval)
4. New Models:
   a. Adam, Ada

PART 3 - Matching Identified Features to MedDRA

Task at this stage: use extracted ADR text from our model and match it to the correct MedDRA concept.

- No MedDRA API



Extraction
space cadet
no immune
neck ache

UMLS
Metamap Term (closest match)
CUI

Dictionary
MedDRA Term
MedDRA Code
CUI

MedDRA Code	**MedDRA Term**
10041374	spaced out
10021425	immune system disorder
10028836	neck pain

Results

We ran F1-Scores comparing the validation file to our predicted results. Below are the results:

- F1-Score for correctly predicting ADR (yes): 0.59
- F1-Score for correctly predicting the extraction text: 0.21
    - F1-Score calculated using a dummy variable
- F1-Score for correctly predicting the MedDRA code: 0.08
    - F1-Score calculated using a dummy variable

Limitations

1. When performing the twitter texts cleaning , we didn’t use the lemmatization technique because it will make the tracing back of ADR span in original twitter more complicated.
2. We used regular expression to find the matched ADR with our predicted extractions in original twitters, the matching accuracy were highly dependent on how the twitter cleaning process changed the words and the regular expression features we used.
3. Due to lack of understanding on the working of the model, we faced a hard time tuning it to get a good f1-score for the extracted ADRs.
4. The data itself is quite messy. Many items with ADRs are not classified as ADRs
    a. “... i ocd worry. can't stop running mind”
5. The quick UMLS matcher had a hard time recognizing less straightforward extractions
    a. “Space Cadet” → “Spaced out”

Future Considerations

1. It would be better if we can mark the position of each word in original twitter and then trace it back after their labels were predicted so that lemmatization could be use. Also, different regular expression features should be tried to increase the matching accuracy.
2. Train the model to “overpredict” ADRs since there are many ADRs that were not classified as such.
3. We would have liked to train a corpus using tools like Tensorflow to create a better semantic match between extractions and MedDRA terms.
    a. The combination of Quick UMLS and our MedDRA-Metamap dictionary lead to a significant loss of accuracy

Questions & Answers