Symptom Extraction and Linking from Vaccine Adverse Event Reports

Background and the Goal:

Sequence labeling has been one of the most well-known topics in linguistics and computational linguistics history. Some examples of sequence labeling include part of speech (POS) tagging, named entity recognition (NER), and event detection (ED). In this project, you will apply sequence labeling techniques to a healthcare data, namely VAERS data, from Vaccine Adverse Events Reporting Systems (VAERS). The goal of the project is to automatically identify **symptoms** after vaccination (i.e., **vaccine adverse events**) from VAERS reports.

Goal: Given a VAERS report, you will

1. Use existing or develop new named entity recognition packages to **identify symptom-related terms from narrative text reports**.
2. Develop named entity linking methods to **link the identified terms to standard terms** in a dictionary.

This document provides the task description along with some suggestions about the methods. The project is designed for you to get a hands-on experience to utilize what we learnt in class to solve the real-application problems. There is no fixed solution for the task. When you work on this project, you don't need to strictly follow the suggested methods. You are encouraged to design, explore, and investigate any methods and packages that can be used to solve the task. Also, feel free to utilize any evaluation methods that are reasonable. For example, one option is that you can consider solve the two tasks in STEP 1 and STEP 2 together using GPT. Specifically, you can design prompts to extract the symptoms and link to the standard terms.

Dataset:

The Vaccine Adverse Event Reporting System (VAERS) is a national early warning system to detect possible safety problems in U.S.-licensed vaccines. VAERS is co-managed by the Centers for Disease Control and Prevention (CDC) and the U.S. Food and Drug Administration (FDA). VAERS accepts and analyzes reports of adverse events (possible side effects) after a person has received a vaccination.

Tasks:

1. **Download Data**

   There are **three tables** in the VAERS dataset, including VAERS DATA, VAERS Symptoms, and VAERS Vaccine.

   - You can find and download the data in VAERS data source link.

- Please read more details about the data in the VAERS data use guide.
- You can also explore the data in The LEAF VAERS Data Analysis System

To prepare the data you use in this project, you will need to (1) find symptom texts from VAERS DATA table, (2) use the VAERS Symptoms table for evaluation, and (3) use the VAERS Vaccine table to filter the data.

2. **Process Data**

Since the VAERS data has more than 1 million records, you need to select a small sample of the dataset. In this project, you can

a. Select a vaccine type based on VAERS Vaccine table. For example, COVID19, FLU3, or TDAP.
b. Based on the VAERS_ID, get around 10,000 reports related to the vaccine you selected from the column SYMPTOM_TEXT in the VAERS DATA table.
c. Perform some statistical analysis of your data. For example, you can get the distribution of the number of different symptoms related that vaccine type. You can also get the distribution of the length of different reports.
d. For the reports you selected in 2.b, build the corresponding standard symptom list with the symptoms in VAERS Symptoms table. Further, please find the most common symptoms (e.g., the most frequent 100 symptoms.).

STEP 1: Extracting Symptom-related Entities:

Task:

- **Input**: Description of vaccine adverse events (i.e., SYMPTOM TEXT in VAERS DATA table).
- **Output**: A list of symptom related entities.
- **Evaluation**: There is no ground truth annotation for the data. You need to manually check 20+ reports to see if there are entities that are missed by the model or package.

Method and implementation:

It is highly recommended to use **Stanza package** for this task. Stanza covers the biomedical and clinical NER models. For your project, you can consider using any model that can extract symptoms related (labeled as PROBLEM or DISEASES in Stanza output) entities for you. For example, **i2b2** package can find entities related to PROBLEM, TEST, and TREATMENT. PROBLEM entities are symptoms. Please feel free to try other packages that identify DISEASE related entities, which can be considered as symptoms.

For more details of the models, please check the original paper about "Biomedical and clinical English model packages for the Stanza Python NLP library". To learn how to install and use Stanza, please check the Stanza download and usage page. You can also check the download and usage

of biomedical models here. In the implementation, you can follow the pipeline with the MIMIC syntactic analysis models and the i2b2 clinical NER model.

Two examples are provided below to show the input and the corresponding outputs.

Example 1:

- **Input:** The adverse event is that the patient went into a coma state and was non responsive. Patient spent almost a month hospitalized and transferred into a nursing home. Trauma to the head caused severe orthostatic blood pressure problems, high fall risk, ongoing headaches, and caused patient to be exposed to covid, Be advised patient was tested the day before with a PCR 3 day covid test that resulted in zero antibodies.
- **Output:**
    - a coma state PROBLEM
    - non responsive PROBLEM
    - Trauma PROBLEM
    - severe orthostatic blood pressure problems PROBLEM
    - high fall risk PROBLEM
    - ongoing headaches PROBLEM
    - a PCR TEST
    - covid test TEST
    - zero antibodies PROBLEM

Example 2:

- **Input:** Given Tdap injection on 1/21/11 and several days later noticed redness and swelling.
- **Output:**
    - Tdap injection TREATMENT
    - redness PROBLEM
    - swelling PROBLEM

In addition to Stanza, you are also encouraged to try other packages and methods (such as clinisift https://github.com/cactiML/clinisift).

STEP 2: Link Entities to Standard Symptoms

Task: For each extracted symptom entity, map it to the term in a standard symptom list.

- **Input**: A symptom detected in STEP 1, Vocabulary 1 (a list of standard symptoms), Vocabulary 2 (a list of the most common standard symptoms).
- **Output**: Its standard symptom. For example, fever -> pyrexia, fatigue -> fatigue.
- **Evaluation**: You can consider using different strategies to evaluate your outcome. For example, recall can be used to evaluate the percentage of symptoms that have been missed by your models. Precision can be used to evaluate if your model has identified

many irrelevant or incorrect symptoms. You can also define your own evaluation metrics. The goal is to

- Evaluate the performance based on standard symptom list.
- Evaluate the performance based on the most common standard symptom list.

Make sure you have both automatic evaluation and manual evaluation. Here, for the manual evaluation, you can select 20~50 clinical notes and manually check the results.

Linking Method: In this project, you are free to develop new entity linking models, use existing methods, adopt rule-based methods, and use large language models like ChatGPT to perform the linking task. For example, the following methods are straightforward and easy to implement:

- **Rule-based matching:**
  - Exact matching: A lot of symptoms in SYMPTOM TEXT can be directly mapped to standard symptoms via exact matching.
  - Fuzzy matching: Some symptoms can be mapped to standard symptoms via Fuzzy matching. You can try this package https://pypi.org/project/fuzzywuzzy/.
- **Similarity-based matching**: You can also consider using similarity-based matching to link the raw symptoms and standard symptoms. Here, you can consider use word embedding to embed tokens and then calculate the similarity. You can try GloVe embedding or clinical word embedding (https://github.com/gweissman/clinical_embeddings).

Project submission:

1. Please submit all the codes you used for the project, which include the implementation for data processing, extraction of symptom-related entities, and entity linking. Make sure all the codes are runnable and with sufficient comments.
2. Please submit the data you used for evaluation as the .csv file or .json file. For example,
   a. A file with three columns for each report you selected: vaers_id, original symptom text, list of standard symptoms, list of symptom mentions you extracted in STEP 1.
   b. List of most common symptoms you identified in STEP 1.
   c. For each linking method you used, you will need to provide the linked results, based on which you performed the evaluation.
3. Please organize the results in the final report following the guideline of the final report submission (will be provided before the submission). It is also important to include detailed discussions about the results analysis, your findings based on the results, and potential aspects to improve.
4. For the projects with multiple team members, to ensure every team member has sufficient contributions to the project, you can consider the following options and include a discussion to compare the findings based on your results from different datasets or different models.

a. Prepare multiple datasets by selecting multiple vaccines during data processing and solve the same task on different datasets.
b. Explore multiple methods for STEP 1 and STEP 2.

Selected References:

1. Qi P, Zhang Y, Zhang Y, Bolton J, Manning CD. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations 2020 Jul (pp. 101-108).
2. Yuhao Zhang, Yuhui Zhang, Peng Qi, Christopher D Manning, Curtis P Langlotz, Biomedical and clinical English model packages for the Stanza Python NLP library, *Journal of the American Medical Informatics Association*, Volume 28, Issue 9, September 2021, Pages 1892–1899, https://doi.org/10.1093/jamia/ocab090
3. Chen Q, Peng Y, Lu Z. BioSentVec: creating sentence embeddings for biomedical texts. The 7th IEEE International Conference on Healthcare Informatics. 2019.
4. Gu Y, Tinn R, Cheng H, Lucas M, Usuyama N, Liu X, Naumann T, Gao J, Poon H. Domain-specific language model pretraining for biomedical natural language processing. ACM Transactions on Computing for Healthcare (HEALTH). 2021 Oct 15;3(1):1-23.