Symptom Extraction and Linking from Vaccine Adverse Event Reports

Thanapoom Phatthanaphan
Natural Language Processing, Computer Science Department
Stevens Institute of Technology

Project Member

Thanapoom Phatthanphan

CWID 20011296
Degree Master's Degree in Computer Science

Project's Goal

- **Develop named entity recognition** packages to **identify symptom related terms** from narrative text reports
- **Develop named entity linking methods** to **link the identified terms to standard terms** in a dictionary

Steps of implementation

Import data → **Preprocessing** → **Extracting Symptom-related Entities** → **Linking Entities to Standard Symptoms** → **Evaluation**

Import data
- Download and inport the VAERS dataset

Preprocessing
- Select significant vaccine type
- Statistical analysis
- Find the standard and the most common symptoms

Extracting Symptom-related Entities
- Download Stanza
- Extract the symptom-related entities in each symptom text

Linking Entities to Standard Symptoms
- Link the extracted symptoms to the standard and most common symptoms

Evaluation
- Evaluate the performance of the model to extract symptom (Precision, Recall, Recall)

Dataset

The reports of adverse events after a person has received a vaccination from the Vaccine Adverse Event Reporting System (VAERS).

Public dataset: https://vaers.hhs.gov/data/datasets.html

Report detail dataset

	VAERS_ID	RECVDATE	STATE	AGE_YRS	CAGE_YR	CAGE_MO	SEX	RPT_DATE	SYMPTOM_TEXT	DIED	...	CUR_ILL	HISTORY	PRIOR_VAX	SPLTTYPE	FORM_
0	375646	01/01/2010	FL	28.0	28.0	NaN	F	01/01/2010	AT A FAMILY GET-TOGETHER, I WAS LAUGHING AND I...	NaN	...	NONE.	ASTHMA, LUPUS, HYPOTHYROIDISM.	NaN	NaN	
1	375647	01/01/2010	IL	75.0	75.0	NaN	M	01/01/2010	Red rash developed on lower part of left arm f...	NaN	...	taking Voltaren for pain in right arm - starte...	sulfa allergies	NaN	NaN	
2	375648	01/01/2010	MN	30.0	30.0	NaN	F	01/01/2010	Nausea and vomiting for	NaN		No	allergy to penicillin	NaN	NaN	

Symptom dataset

	VAERS_ID	SYMPTOM1	SYMPTOMVERSION1	SYMPTOM2	SYMPTOMVERSION2	SYMPTOM3	SYMPTOMVERSION3	SYMPTOM4	SYMPTOMVERSION4	SYMPTOM5	SYMPTOMVERSION5
0	375646	Chest X-ray normal	12.1	Chest discomfort	12.1	Cough	12.1	Dyspnoea	12.1	Wheezing	12.1
1	375647	Erythema	12.1	Rash	12.1	Rash macular	12.1	NaN	NaN	NaN	NaN
2	375648	Nausea	12.1	Vomiting	12.1	NaN	NaN	NaN	NaN	NaN	NaN
3	375650	Abdominal pain upper	12.1	Diarrhoea	12.1	Fatigue	12.1	Headache	12.1	Myalgia	12.1
4	375650	Nausea	12.1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Vaccine dataset

	VAERS_ID	VAX_TYPE	VAX_MANU	VAX_LOT	VAX_DOSE_SERIES	VAX_ROUTE	VAX_SITE	VAX_NAME
0	375646	FLU(H1N1)	SANOFI PASTEUR	NaN	1	IM	LA	INFLUENZA (H1N1) (H1N1 (MONOVALENT) (SANOFI))
1	375647	FLU(H1N1)	SANOFI PASTEUR	UP078AA	1	IM	LA	INFLUENZA (H1N1) (H1N1 (MONOVALENT) (SANOFI))
2	375648	FLUX(H1N1)	UNKNOWN MANUFACTURER	NaN	1	NaN	NaN	INFLUENZA (H1N1) (H1N1 (MONOVALENT) (UNKNOWN))
3	375650	FLUX(H1N1)	UNKNOWN MANUFACTURER	NaN	1	IM	LA	INFLUENZA (H1N1) (H1N1 (MONOVALENT) (UNKNOWN))
4	375651	VARZOS	UNKNOWN MANUFACTURER	NaN	1	NaN	RA	ZOSTER (NO BRAND NAME)

Data handling

pandas NumPy

Plotting

matplotlib

python + colab

Stanza

Extracting method

Fuzzy Matching GENSIM topic modelling for humans

Linking methods

1. Importing the libraries

Data handling and plotting

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob
```

Extracting method

```
!pip install stanza

# Import necessary library
import stanza

# download and initialize a mimic pipeline with an i2b2 NER model
stanza.download('en', package='mimic', processors={'ner': 'i2b2'})
nlp = stanza.Pipeline('en', package='mimic', processors={'ner': 'i2b2'})
```

Linking method

```
# Import the libraries for fuzzywuzzy
!pip install fuzzywuzzy
from fuzzywuzzy import fuzz
from gensim.models import KeyedVectors
```

2. Data preprocessing

- Find the largest number of data based on Vaccine type

```
# Get the number of data based on VAX_TYPE for
vaers_vax['VAX_TYPE'].value_counts().head()

COVID19    1009575
VARZOS      108667
FLU3         57602
FLU4         43724
PPV          42530
Name: VAX_TYPE, dtype: int64
```

The largest one is COVID-19
(Select this one for the model)

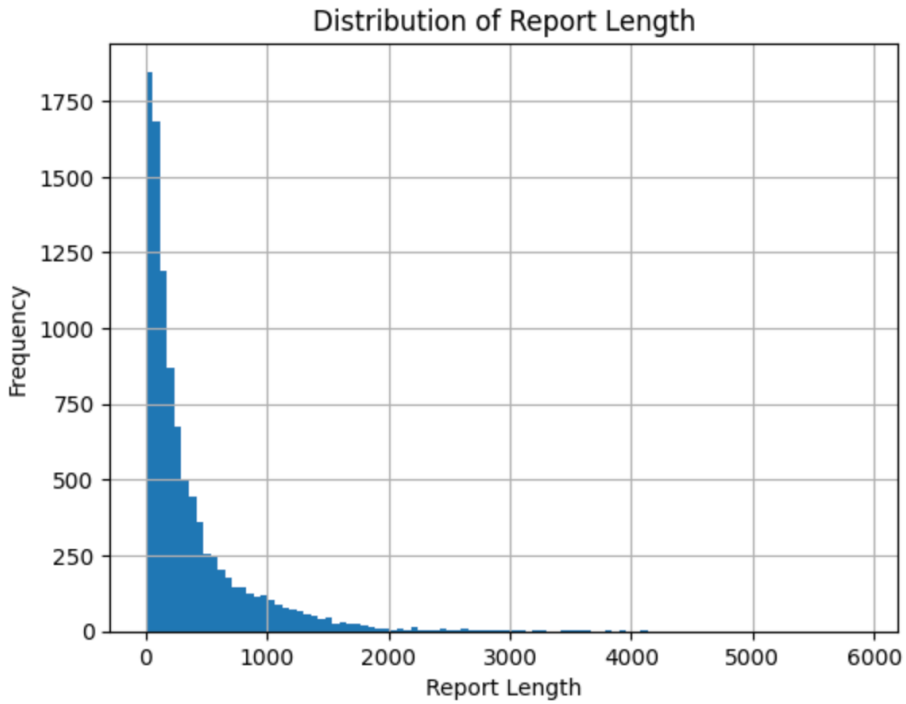
- Select 10,000 reports about COVID-19

	VAERS_ID	SYMPTOM_TEXT
0	2669769	body aches, fatigue Narrative: Took OTC Tyleno...
1	2527460	Headache, Myalgia, NauseaVomiting, chills Narr...
2	2673135	Headache, Fever, Body aches Narrative: Other ...
3	2672717	Headache & Myalgia Narrative: Other Relevant...
4	902418	Patient experienced mild numbness traveling fr...
...	...	...
9995	916173	REDNESS TO INJECTION SITE 12-30-20. PROGRESSED...
9996	916174	Patient described joint and muscle pain in the...
9997	916176	Numbness and tingling on left side of face, ey...
9998	916177	HIVES, tachypnea, vomiting - normal saline, ne...
9999	916178	Excessive swelling to left axillary lymph node...

10000 rows × 2 columns

2. Data preprocessing (Data statistics)

- Plotting the distribution charts to see the length of the symptom text of these 10,000 selected covid-19 reports



Average: 376 characters
Minimum: 2 characters
Maximum: 5904 characters

2. Data preprocessing

- Get the lists of standard symptom and the top 100 common symptoms

Link between 2 dataset using VAERS_ID

	VAERS_ID	SYMPTOM_TEXT
0	2669769	body aches, fatigue Narrative: Took OTC Tyleno...
1	2527460	Headache, Myalgia, NauseaVomiting, chills Narr...
2	2673135	Headache, Fever, Body aches Narrative: Other ...
3	2672717	Headache & Myalgia Narrative: Other Relevant...
4	902418	Patient experienced mild numbness traveling fr...

10,000 selected covid-19 reports

	VAERS_ID	SYMPTOM1	SYMPTOMVERSION1	SYMPTOM2	SYMPTOMVERSION2	SYMPTOM3	SYM
0	375646	Chest X-ray normal	12.1	Chest discomfort	12.1	Cough	
1	375647	Erythema	12.1	Rash	12.1	Rash macular	
2	375648	Nausea	12.1	Vomiting	12.1	NaN	
3	375650	Abdominal pain upper	12.1	Diarrhoea	12.1	Fatigue	
4	375650	Nausea	12.1	NaN	NaN	NaN	

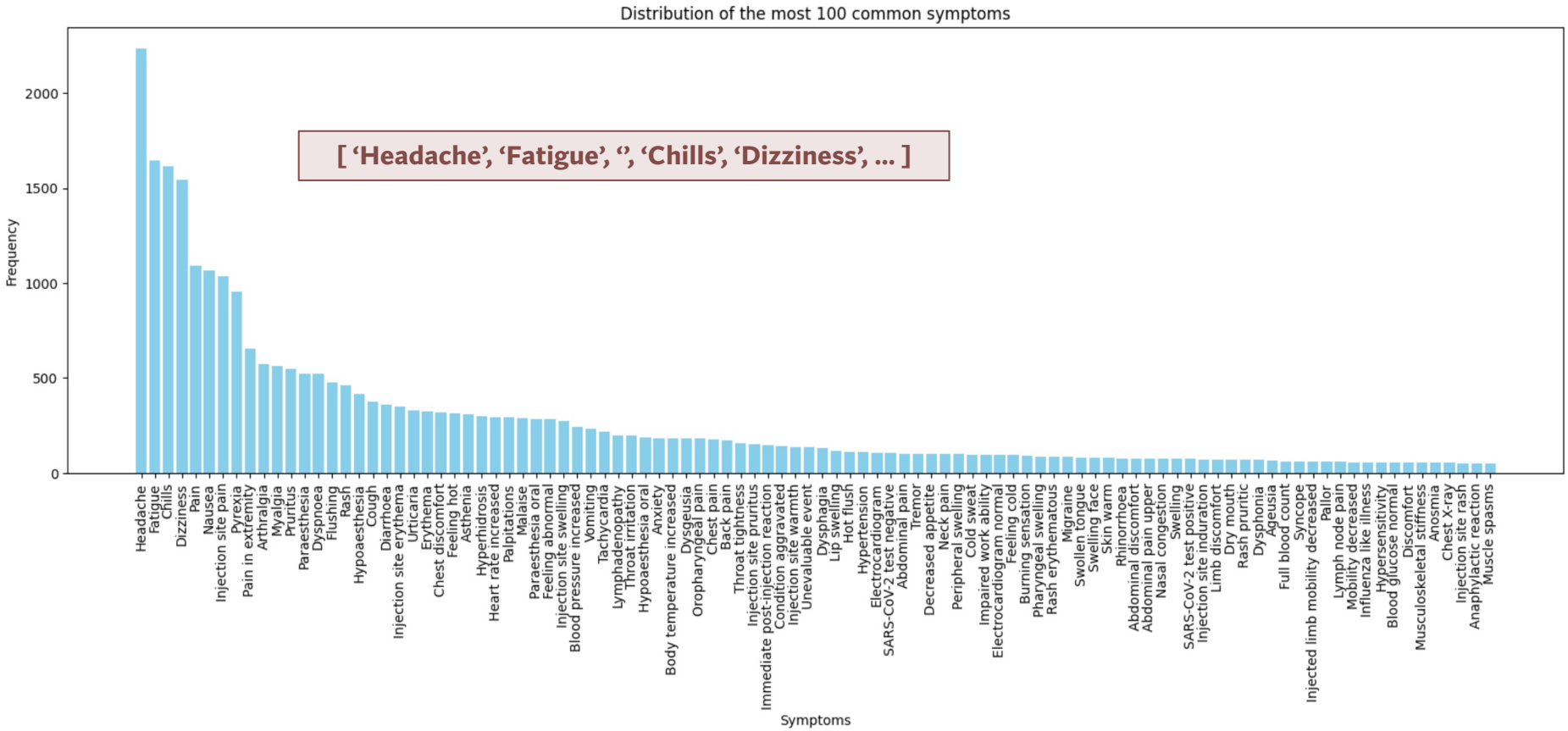
Symptom dataset

Got 2 lists

- **[‘Fatigue’, ‘Pain’, ‘Chills’, ‘Headache’, ‘Myalgia’, ...]** — **All standard symptoms**
- **[‘Headache’, ‘Fatigue’, ‘’, ‘Chills’, ‘Dizziness’, ...]** — **Top 100 common symptoms**

2. Data preprocessing (Data statistics)

- Plotting the distribution charts to see the number of each symptom



3. Extracting Symptom-related entities

- Import stanza and download the English language model with a named entity recognition processor called i2b2 from mimic package.
- Initialize a Stanza pipeline

```
!pip install stanza

# Import necessary library
import stanza

# download and initialize a mimic pipeline with an i2b2 NER model
stanza.download('en', package='mimic', processors={'ner': 'i2b2'})
nlp = stanza.Pipeline('en', package='mimic', processors={'ner': 'i2b2'})
```

Download the model

Initialize a pipeline

Mimic package provides pre-trained models for clinical text processing

3. Extracting Symptom-related entities

- Extract the symptoms from the symptom text of covid-19 reports, then store those symptoms in a list

```
Vaers ID: 2669769
Input: body aches, fatigue Narrative: Took OTC Tylenol  Other Relevant History:
Output:
        body aches      PROBLEM
        fatigue PROBLEM
        OTC Tylenol     TREATMENT

Vaers ID: 2527460
Input: Headache, Myalgia, NauseaVomiting, chills Narrative
Output:
        Headache        PROBLEM
        Myalgia PROBLEM
        NauseaVomiting  PROBLEM
        chills  PROBLEM

Vaers ID: 2673135
Input: Headache, Fever, Body aches Narrative:  Other Relevant History:
Output:
        Headache        PROBLEM
        Fever   PROBLEM
        Body aches      PROBLEM
```

Store these problems (symptoms) in a list

```
['body aches',
 'fatigue',
 'headache',
 'myalgia',
 'nauseavomiting',
 'chills',
 'headache',
 'fever',
 'body aches',
 'headache',
 'myalgia',
 'mild numbness',
 'injection site',
 'headache',
 'warm',
 'progressive light-headedness',
 'near-syncope',
 'diaphoresis',
 '20 minutes symptoms'
```

4. Linking Entities to the standard symptoms

- Link the extracted symptoms to the standard symptoms and the top 100 common symptoms, using **Rule-based (Exact matching and Fuzzy matching), and Similarity-based matching.**
  (Note: Testing only 50 reports for not too long running time)

Method 1: Exact matching

```python
# Define the dictionary for mapping
exact_linked_std_symp_dict = {}
exact_linked_top_symp_dict = {}

# Iterate to find the exact match symptom
for symp in extracted_symptom_list:

    # Define as None at first
    exact_linked_std_symp_dict[symp.lower()] = 'None'
    exact_linked_top_symp_dict[symp.lower()] = 'None'

    # Iterate to find the exact match symptom from the list of standard symptoms
    for std_symptom in std_symp:
        if symp.lower() == std_symptom.lower():
            exact_linked_std_symp_dict[symp.lower()] = std_symptom.lower()
            break

    # Iterate to find the exact match symptom from the list of top 100 symptoms
    for top_symptom in top_symp:
        if symp.lower() == top_symptom.lower():
            exact_linked_top_symp_dict[symp.lower()] = top_symptom.lower()
            break
```

```
{'body aches': 'None',
 'fatigue': 'fatigue',
 'headache': 'headache',
 'myalgia': 'myalgia',
 'nauseavomiting': 'None',
 'chills': 'chills',
 'fever': 'None',
 'mild numbness': 'None',
 'injection site': 'None',
 'warm': 'None',
 'progressive light-headedness': 'None',
 'near-syncope': 'None',
 'diaphoresis': 'None',
 '20 minutes symptoms': 'None',
```

Link the extracted symptom to the standard symptom that is **exactly matching**.

4. Linking Entities to the standard symptoms

Method 2: Fuzzy matching

```
# Iterate to find the most similar symptom from the list of standard symptoms
max_fuzz_ratio = 0.0
for std_symptom in std_symp:
    fuzz_ratio =  fuzz.partial_ratio(symp.lower(), std_symptom.lower())
    if fuzz_ratio > max_fuzz_ratio:
        max_fuzz_ratio = fuzz_ratio
        most_sim_std_symp = std_symptom.lower()
fuzzy_linked_std_symp_dict[symp.lower()] = most_sim_std_symp
```

```
{'body aches': 'acne',
 'fatigue': 'fatigue',
 'headache': 'headache',
 'myalgia': 'myalgia',
 'nauseavomiting': 'nausea',
 'chills': 'chills',
 'fever': 'thyroxine free',
```

Link the extracted symptom to the standard symptom with **the most similar sequence of characters**

Method 3: Similarity-based matching

```
# Find the most similar symptoms from the list of standard symptoms
max_sim_score = 0.0
most_symp = 'None'
for std_symptom in std_symp:

    std_symptom_vec = get_embedding(std_symptom.lower(), glove_embeddings, default_vector)
    sim_score = np.dot(symp_vec, std_symptom_vec) / (np.linalg.norm(symp_vec) * np.linalg.norm(std_symptom_vec))
    if sim_score > max_sim_score:
        max_sim_score = sim_score
        most_symp = std_symptom

sim_linked_std_symp_dict[symp] = most_symp
```

```
{'body aches': 'None',
 'fatigue': 'Fatigue',
 'headache': 'Headache',
 'myalgia': 'Myalgia',
 'nauseavomiting': 'None',
 'chills': 'Chills',
```

Get the vector of each symptom using **GloVe**, then **find the most similar** standard symptom using **Cosine-similarity**

5. Evaluation (Automatic)

- Evaluate the model by computing cosine similarity score between the extracted symptoms and the standard symptoms.

Criteria

- Similarity score >= 0.8 : True positive
- Similarity score < 0.8 : False positive
- Not found : False negative
  (#Standard symptoms - #Extracted symptoms)

```
{'body aches': 'acne',
 'fatigue': 'fatigue',
 'headache': 'headache',
 'myalgia': 'myalgia',
 'nauseavomiting': 'nausea',
 'chills': 'chills',
 'fever': 'thyroxine free',
```

Linking methods	Comparison	Precision	Recall	F1 score
Exact matching	Standard	0.179	0.394	0.246
	Top 100 common	0.174	0.387	0.240
Fuzzy matching	Standard	0.179	0.394	0.246
	Top 100 common	0.174	0.387	0.240
Cosine-similarity matching	Standard	0.237	0.462	0.313
	Top 100 common	0.217	0.441	0.291

Best results

(Note: Testing only 50 reports for not too long running time)

5. Evaluation (Manual)

- Manually check each report for 20 reports

```
Report No.: 1
Vaers ID: 2669769
Input: body aches, fatigue Narrative: Took OTC Tylenol  Other Relevant History:
Symptoms:
        body aches
        fatigue

Standard symptoms:
         Fatigue
         Pain
```

#Reports: 20 reports
#Extracted all symptoms: 13 reports
The model can completely extract symptoms for 13 reports (65%)

#Standard symptoms: 68 symptoms
#Extracted symptoms: 60 symptoms
The model can extract the symptoms from 20 reports for 79.41%

```python
# Finding those misses symptoms and found symptoms
missed_symptom = ['Erythema', 'Flushing', 'Blood pressure increased', 'Visual impairment', 'Eye pruritus', 'Asthenia', 'Heart rate increased', 'Hypertens
found_symptom = 0
for symp in std_symptom_20:
    if symp not in missed_symptom:
        found_symptom += 1

# Calculate the percentage of discovered symptoms from 20 reports
total_std_symptom_20 = len(std_symptom_20)
percentage_found_symptoms = (found_symptom / total_std_symptom_20) * 100
print(f"The percentage of discovered symptoms from 20 report, compared to the standard symptoms: {percentage_found_symptoms} %")
print("Missed symptoms:")
for i, symp in enumerate(missed_symptom):
    print(f"{i + 1}. {symp}")
```

```
The percentage of discovered symptoms from 20 report, compared to the standard symptoms: 79.41176470588235 %
Missed symptoms:
1. Erythema
2. Flushing
3. Blood pressure increased
4. Visual impairment
5. Eye pruritus
6. Asthenia
7. Heart rate increased
8. Hypertension
```

Missing symptoms that the model could not extract (Found from manually checking)

Thank you

for your attention