II. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

   I am a data scientist at a Japanese tech company. My current role involves promoting and facilitating the efficient use of ChatGPT in business, as well as exploring innovative applications of generative AI. I have 2 years of experience in data analysis and over 3 years of experience in developing deep learning architectures. Outside of work, I enjoy traveling and comedy shows.

2. What motivated you to compete in this challenge?

   During an actual data analysis project that I was involved in, I had the opportunity to extract insights from a large-scale text dataset similar to what we used for this project. I wanted to apply the knowledge I gained from that experience to other data science projects and also developed an interest in exploring new ways to utilize Large Language Models (LLMs). I'm eager to apply the insights I've gained here to our ongoing projects.

3. High level summary of your approach: what did you do and why?

   I believed that there is a preceding action that caused the fall as information unique to Narrative. While there is information about the object that caused the fall, it alone cannot infer the fundamental cause of the fall, leading to devising fall prevention measures. Hence, I focused on the information, "What action led to the fall?”. As my approach to get the information, I carried out the extraction of actions by ChatGPT using the OpenAI API. And with this result, I analyzed Primary data to try to gain some insights with other columns.

4. Do you have any useful charts, graphs, or visualizations from the process?

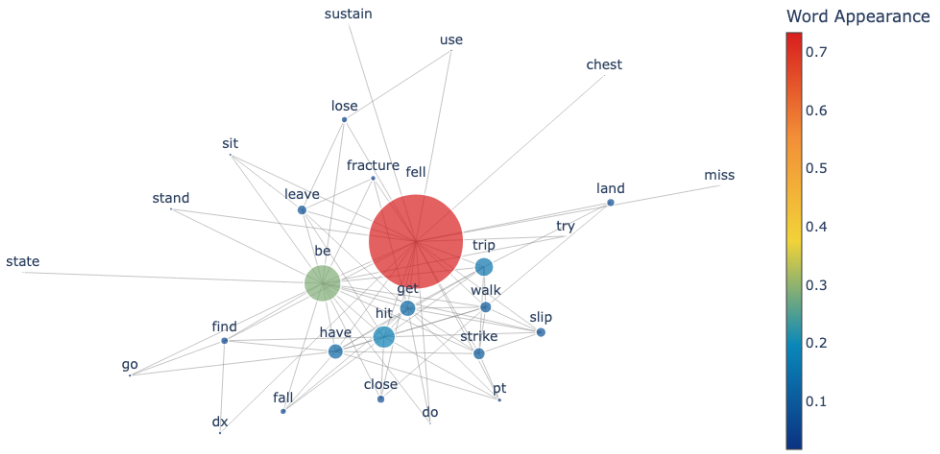

Figure 4: Co-occurrence network based on action information extraction results from OpenAI API.

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.
   - I used Few-shot prompting to extract actions.
     By giving some examples, I could tune and get the outputs as I expected.

```
examples = [
    {
        "text": '94YOM FELL TO THE FLOOR AT THE NURSING HOME ONTO
BACK OF HEAD SUSTAINED A SUBDURAL HEMATOMA',
        "verb": "FELL TO THE FLOOR"
    },
    {
        "text": '87YOF WAS GETTING UP FROM THE COUCH AND FELL TO THE
FLOOR SUSTAINED ABRASIONS TO ELBOWS ADMITTED FOR
HEMORRHAGIC STROKE',
        "verb": "GETTING UP FROM THE COUCH"
    },
    {
        "text": '67YOF WAS AT A FRIENDS HOUSE AND  THAT WAS ON THE
FLOOR AND SUSTAINED A RIGHT RADIUS FX',
        "verb": "SLIPPED ON WATER"
    },
]

example_formatter_template = """
text: {text}
verb: {verb}\n
"""
example_prompt = PromptTemplate(
    template=example_formatter_template,
    input_variables=["text", "verb"]
)
```

```
few_shot_prompt = FewShotPromptTemplate(
    examples=examples,
    example_prompt=example_prompt,
    prefix="These are sentences describing situations when elderly individuals fell. Please extract from the sentences the causes of the falls and the actions the elderly individuals took when they fell.",
    suffix="text: {input}\nverb:",
    input_variables=["input"],
    example_separator="\n\n",

)
```

This code is to draw my favorite co-occurrence network. This is beautiful and easy to grasp the relationships among words and beautiful. Let me share the code, here: https://gist.github.com/RxstydnR/f251f8903a300d7c9fcfc04496cad419

```
# ネットワークの構築
def build_interactive_network(G, pos, node_sizes, node_colors):

    # edgeデータの作成
    edge_x = []
    edge_y = []
    for edge in G.edges():
        x0, y0 = pos[edge[0]]
        x1, y1 = pos[edge[1]]
        edge_x.append(x0)
        edge_x.append(x1)
        edge_x.append(None)
        edge_y.append(y0)
        edge_y.append(y1)
        edge_y.append(None)

    # edgeデータの描画
    edge_trace = go.Scatter(
        x=edge_x, y=edge_y,
        line=dict(width=0.5,color='#888'),
        # hoverinfo='none',
        opacity=1,
        mode='lines')

    # nodeデータの作成
    node_x = []
    node_y = []
    for node in G.nodes():
        x, y = pos[node]
        node_x.append(x)
        node_y.append(y)

    # ホバー時の表示テキスト作成
    hover_texts = [f"{word}:{val}" for word,val in zip(list(G.nodes()),node_sizes)]

    # nodeの色、サイズ、マウスオーバーしたときに表示するテキストの設定
    node_trace = go.Scatter(
```

```
        x=node_x,
        y=node_y,
        text=list(G.nodes()),
        hovertext=hover_texts,#node_sizes,
        textposition='top center',
        mode='markers+text',
        hoverinfo='text',
        marker=dict(
            showscale=True,
            colorscale='Portland',
            reversescale=False,
            color=node_colors,
            size=norm_node_sizes(node_sizes), # size=node_sizes,
            colorbar=dict(
                thickness=16,
                title='Word Appearance',
            ),
            line_width=2))

    data = [edge_trace, node_trace]

    # レイアウトの設定
    layout=go.Layout(
                paper_bgcolor='rgba(0,0,0,0)',
                plot_bgcolor='rgba(0,0,0,0)',
                showlegend=False,
                hovermode='closest',
                # margin=dict(b=10, l=5, r=5, t=10),
                margin=dict(b=10, l=5, r=5, t=0),
                font=dict(size=14),
                xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
                yaxis = dict(showgrid = False, zeroline = False, showticklabels = False))

    fig = go.Figure(data=data, layout=layout)
    fig.update_layout(
        xaxis_title=None,
        # title=dict(text=f"頻出単語を表示",font=dict(size=30)),
        dragmode='pan',
        margin=dict(b=0,l=0,r=0,t=20),
        # title_pad=dict(b=0,l=10,r=0,t=15),
        autosize=False, width=1000, height=500
    )
    # fig.show()
    fig.show(config=plotly_config)
```

Using a code below, I could extract action information without noise. It helped me to get clear insights.

```
from nltk.corpus import stopwords
# Get each word list of narratives
# Remove unnecessary words from narrative
remove_pos_tag_list = ['CC', 'JJ', ',', '.', ')', '(', 'IN']
```

```
stop_words = stopwords.words('english')
stop_words += ["sustained","was","fx","wa"]

def remove_words_from_narrative(s):
    s = s.lower()
    ws = nltk.word_tokenize(s)
    ws = [w for w in ws if w.isalpha()]
    pos = nltk.pos_tag(ws)
    ws = [p[0] for p in pos if (not p[1] in remove_pos_tag_list)&(not p[0] in
stop_words)]
    return ws

df_ret['ChatGPT_refined'] =
df_ret['ChatGPT'].map(remove_words_from_narrative)
lines = df_ret['ChatGPT_refined'].values
get_cooccurrence_network(lines, edge_threshold=3, freq_top_n=30)
```

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): 2GHz quad core Intel Core i5
   - GPU (model or N/A): N/A
   - Memory (GB): 32 GB
   - OS: MacOS 13.4.1
   - Train duration: No training
   - Inference duration: About 30 min to extract actions from 300 texts. (But I know now I can shorten the time)

7. Anything we should watch out for or be aware of in using your notebook (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

   You have to use OpenAI API as described in my executive summary of midpoint submissions. Needless to say but it may cost you much.

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

   No, I did not. I only used the codes in my jupyter notebook.

9. How did you evaluate the quality of your analysis and insights, if at all?

   I evaluated the quality of the analysis and its results based solely on subjective qualitative assessment.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

    There aren't any such things.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

    I couldn't come up with good ideas on how to properly use models like BERT or extract insights using the models. I want to use them to attempt innovative challenges on this problem. And also I am interested in Knowlege-Graph. As cooccurrence network was working well to understand the preceding actions that caused the fall, I believe Knowlege-Graph has a power to organize the Narrative information.

12. What simplifications could be made to run your solution faster without sacrificing performance?

    The most time-consuming part of my approach is extracting actions from the narrative by ChatGPT. It has already been proven that this can be sped up through asynchronous processing.