```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "6954adf1-8ca6-4abc-ab69-ec363bede46d",
   "metadata": {},
   "outputs": [],
   "source": [
    "import json\n",
    "import numpy as np\n",
    "import pandas as pd\n",
    "import random\n",
    "import re\n",
    "import time\n",
    "from pathlib import Path\n",
    "\n",
    "import os\n",
    "import openai\n",
    "from openai import OpenAI\n",
    "import tiktoken\n",
    "\n",
    "import matplotlib as plt\n",
    "import seaborn as sns\n",
    "\n",
    "pd.set_option('display.max_colwidth', None)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "id": "2fd3103a-98ef-486e-a84e-9ff2c552a519",
   "metadata": {},
   "outputs": [],
   "source": [
    "# OPEN AI KEY\n",
    "client = OpenAI(api_key='your key')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "id": "edfa6822-98ba-421d-ab88-8f485a3d8217",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Loading data\n",
    "pol = Path(r\"C:\\Users\\ASUS\\Desktop\\UW\\Winter 2024\\LING 575 D\\A6\\data\\data\\test\\us_foreign_policy_t
    "df_pol = pd.read_csv(pol, names=['question', 'A', 'B', 'C', 'D', 'answer'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "id": "53e26edc-b7cf-4510-bdb1-900258cddcb9",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
```

```
 "      <th></th>\n",
 "      <th>question</th>\n",
 "      <th>A</th>\n",
 "      <th>B</th>\n",
 "      <th>C</th>\n",
 "      <th>D</th>\n",
 "      <th>answer</th>\n",
 "    </tr>\n",
 "  </thead>\n",
 "  <tbody>\n",
 "    <tr>\n",
 "      <th>0</th>\n",
 "      <td>What is the structure of the United Nations Security Council?</td>\n",
 "      <td>5 permanent members with veto power, 10 rotating members with no veto power</td>\n",
 "      <td>5 permanent members and 10 rotating members, all with veto power</td>\n",
 "      <td>10 permanent members with veto power, and 5 rotating members without veto power</td>\n",
 "      <td>15 permanent members with veto power</td>\n",
 "      <td>A</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>1</th>\n",
 "      <td>What was the significance of the Gulf of Tonkin resolution?</td>\n",
 "      <td>It allowed the US to intensify its involvement in Vietnam</td>\n",
 "      <td>It illustrated the influence of public opinion on US foreign policy</td>\n",
 "      <td>It enhanced Congressional control over the Vietnam War</td>\n",
 "      <td>It curtailed US involvement in Vietnam</td>\n",
 "      <td>A</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>2</th>\n",
 "      <td>Which is not a nonstate actor that poses a threat to the United States?</td>\n",
 "      <td>Terrorists</td>\n",
 "      <td>Organized crime</td>\n",
 "      <td>Drug traffickers</td>\n",
 "      <td>China</td>\n",
 "      <td>D</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>3</th>\n",
 "      <td>Who was the first American president to visit communist China?</td>\n",
 "      <td>Richard Nixon</td>\n",
 "      <td>George H. W. Bush</td>\n",
 "      <td>Jimmy Carter</td>\n",
 "      <td>Ronald Reagan</td>\n",
 "      <td>A</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>4</th>\n",
 "      <td>The Strategic Arms Reduction Treaty was the first accord</td>\n",
 "      <td>on nuclear weapons signed between the United States and the Soviet Union.</td>\n",
 "      <td>cutting conventional arms in Europe.</td>\n",
 "      <td>to be rejected by the U.S. Senate.</td>\n",
 "      <td>mandating the elimination of many long-range nuclear missiles.</td>\n",
 "      <td>D</td>\n",
 "    </tr>\n",
 "  </tbody>\n",
 "</table>\n",
 "</div>"
],
"text/plain": [
 "                                                                question  \\\n",
 "0            What is the structure of the United Nations Security Council?   \n",
 "1              What was the significance of the Gulf of Tonkin resolution?   \n",
 "2  Which is not a nonstate actor that poses a threat to the United States?   \n",
 "3           Who was the first American president to visit communist China?   \n",
 "4                The Strategic Arms Reduction Treaty was the first accord   \n",
 "\n",
 "                                                                          A  \\\n",
 "0  5 permanent members with veto power, 10 rotating members with no veto power   \n",
 "1                   It allowed the US to intensify its involvement in Vietnam   \n",
 "2                                                                Terrorists   \n",
 "3                                                              Richard Nixon   \n",
 "4    on nuclear weapons signed between the United States and the Soviet Union.   \n",
 "\n",
 "                                                                          B  \\\n",
 "0     5 permanent members and 10 rotating members, all with veto power   \n",
 "1  It illustrated the influence of public opinion on US foreign policy   \n",
```

```
       "2                                                  Organized crime   \n",
       "3                                                George H. W. Bush   \n",
       "4                                cutting conventional arms in Europe.   \n",
       "\n",
       "                                                                                C  \\\n",
       "0  10 permanent members with veto power, and 5 rotating members without veto power   \n",
       "1                          It enhanced Congressional control over the Vietnam War   \n",
       "2                                                        Drug traffickers   \n",
       "3                                                            Jimmy Carter   \n",
       "4                                          to be rejected by the U.S. Senate.   \n",
       "\n",
       "                                                              D answer  \n",
       "0                          15 permanent members with veto power      A  \n",
       "1                        It curtailed US involvement in Vietnam      A  \n",
       "2                                                      China      D  \n",
       "3                                                Ronald Reagan      A  \n",
       "4  mandating the elimination of many long-range nuclear missiles.      D  "
      ]
     },
     "execution_count": 4,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "# Quick check / Visualization\n",
    "df_pol.head()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "id": "20f7f25c-4379-4a9c-9d1f-37e915a81f1b",
   "metadata": {},
   "outputs": [],
   "source": [
    "#creating a base-line prompts:\n",
    "def make_prompts(df):\n",
    "    \"\"\"\n",
    "    \"\"\"\n",
    "    prompts = []\n",
    "    system_instruct = {\"role\": \"system\", \"content\": \"Your task is to answer the following US-Foreign Pc\n",
    "    \n",
    "    \n",
    "    for index, row in df.iterrows():\n",
    "        prompt_content = f\"{row['question']}\\nA: {row['A']}\\nB: {row['B']}\\nC: {row['C']}\\nD: {row['D']}\\\n",
    "        user_instruct = {\"role\": \"user\", \"content\": prompt_content}\n",
    "\n",
    "        prompt_list = [system_instruct, user_instruct]\n",
    "        prompts.append(prompt_list)\n",
    "    return prompts"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "id": "cdd45bf7-bcc7-490b-8d91-09ffc9bca299",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[{'role': 'system', 'content': 'Your task is to answer the following US-Foreign Policy question by picking t\n"
     ]
    }
   ],
   "source": [
    "base_prompts = make_prompts(df_pol)\n",
    "print(base_prompts[1])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "id": "424172eb-6944-4ce3-b318-321429ce5564",
   "metadata": {},
```

```
 "outputs": [],
 "source": [
  "# Experiment with baseline\n",
  "MODEL = 'gpt-3.5-turbo'\n",
  "encoding = tiktoken.encoding_for_model(MODEL)\n",
  "dict_logit_bias = {}\n",
  "excluded = ['The', 'Answer', 'Correct', 'None']\n",
  "for e in excluded:\n",
  "    dict_logit_bias[encoding.encode(e)[0]]=-100\n",
  "\n",
  "for j in encoding.encode(\"A B C D\"):\n",
  "    dict_logit_bias[j]=-0.01"
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "id": "99a26598-8ddf-4ef7-b793-3880eea622b7",
 "metadata": {},
 "outputs": [],
 "source": [
  "def send_prompt(MESSAGES):\n",
  "    MAX_NEW_TOKENS=1\n",
  "    # flattened_messages = [msg for sublist in MESSAGES for msg in sublist]\n",
  "    \n",
  "    completion = client.chat.completions.create(\n",
  "          model=MODEL,\n",
  "          messages=MESSAGES,\n",
  "          temperature=1.0,\n",
  "          max_tokens=MAX_NEW_TOKENS,\n",
  "          frequency_penalty=0.0,\n",
  "          presence_penalty=0.0,\n",
  "          logprobs=True,\n",
  "          top_logprobs=3,\n",
  "          logit_bias = dict_logit_bias\n",
  "        )\n",
  "\n",
  "    # return both content (ABCD) and its log probabilities\n",
  "    answer = completion.choices[0].message.content\n",
  "    log_prob = completion.choices[0].logprobs.content[0].top_logprobs\n",
  "    return answer, log_prob"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "id": "eec81531-aeda-427c-95ae-a3954450f775",
 "metadata": {
  "scrolled": true
 },
 "outputs": [],
 "source": [
  "# sample test with 25 base examples\n",
  "results = []\n",
  "def run_experiments(PROMPTS, results):\n",
  "    for prompt in PROMPTS[:25]:\n",
  "        # print(\"****\"*8)\n",
  "        # print()\n",
  "        \n",
  "        messages = prompt\n",
  "        # print(messages)\n",
  "        # print()\n",
  "        \n",
  "        final_answer, log_prob = send_prompt(messages)\n",
  "        # print(f\"Model answer: {final_answer}\")\n",
  "        log_probs_formatted = \", \".join([f\"{token.token}: {token.logprob:.4f}, {round(np.exp(token.logprob)\n",
  "        # print(f\"Log Probabilities: {log_probs_formatted}\")\n",
  "\n",
  "        results.append((final_answer, log_prob))\n",
  "    return results\n",
  "# run_experiments(base_prompts, results)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 10,
 "id": "ba6b93ad-ab89-4541-8c4f-e01022c9481e",
```

```
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "[('A',\n",
      "  [TopLogprob(token='A', bytes=[65], logprob=-0.00023202639),\n",
      "   TopLogprob(token=' A', bytes=[32, 65], logprob=-8.691209),\n",
      "   TopLogprob(token='Choice', bytes=[67, 104, 111, 105, 99, 101], logprob=-9.912509)]),\n",
      " ('A',\n",
      "  [TopLogprob(token='A', bytes=[65], logprob=-4.012684e-05),\n",
      "   TopLogprob(token=' A', bytes=[32, 65], logprob=-11.115381),\n",
      "   TopLogprob(token='Choice', bytes=[67, 104, 111, 105, 99, 101], logprob=-11.22123)]),\n",
      " ('D',\n",
      "  [TopLogprob(token='D', bytes=[68], logprob=-0.00032354548),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-8.368409),\n",
      "   TopLogprob(token='A', bytes=[65], logprob=-10.562709)]),\n",
      " ('A',\n",
      "  [TopLogprob(token='A', bytes=[65], logprob=-9.8536635e-05),\n",
      "   TopLogprob(token=' A', bytes=[32, 65], logprob=-9.522891),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-11.868707)]),\n",
      " ('A',\n",
      "  [TopLogprob(token='A', bytes=[65], logprob=-7.827201e-05),\n",
      "   TopLogprob(token='D', bytes=[68], logprob=-10.29447),\n",
      "   TopLogprob(token=' A', bytes=[32, 65], logprob=-10.433613)]),\n",
      " ('D',\n",
      "  [TopLogprob(token='D', bytes=[68], logprob=-0.00075816945),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-7.9949884),\n",
      "   TopLogprob(token='B', bytes=[66], logprob=-8.655953)]),\n",
      " ('A',\n",
      "  [TopLogprob(token='A', bytes=[65], logprob=-0.0064049624),\n",
      "   TopLogprob(token='B', bytes=[66], logprob=-5.3633404),\n",
      "   TopLogprob(token='Choice', bytes=[67, 104, 111, 105, 99, 101], logprob=-6.9756966)]),\n",
      " ('D',\n",
      "  [TopLogprob(token='D', bytes=[68], logprob=-0.012725543),\n",
      "   TopLogprob(token='B', bytes=[66], logprob=-4.431577),\n",
      "   TopLogprob(token='A', bytes=[65], logprob=-7.4311233)]),\n",
      " ('D',\n",
      "  [TopLogprob(token='D', bytes=[68], logprob=-3.166338e-05),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-11.057549),\n",
      "   TopLogprob(token=' D', bytes=[32, 68], logprob=-11.879546)]),\n",
      " ('D',\n",
      "  [TopLogprob(token='D', bytes=[68], logprob=-0.00021164624),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-9.294373),\n",
      "   TopLogprob(token='A', bytes=[65], logprob=-9.634537)]),\n",
      " ('B',\n",
      "  [TopLogprob(token='B', bytes=[66], logprob=-0.00038729745),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-8.567085),\n",
      "   TopLogprob(token='Choice', bytes=[67, 104, 111, 105, 99, 101], logprob=-9.173973)]),\n",
      " ('B',\n",
      "  [TopLogprob(token='B', bytes=[66], logprob=-0.21114652),\n",
      "   TopLogprob(token='C', bytes=[67], logprob=-1.7465602),\n",
      "   TopLogprob(token='D', bytes=[68], logprob=-4.4578276)]),\n",
      " ('B',\n",
      "  [TopLogprob(token='B', bytes=[66], logprob=-8.590105e-05),\n",
      "   TopLogprob(token='According', bytes=[65, 99, 99, 111, 114, 100, 105, 110, 103], logprob=-10.254763),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-10.917482)]),\n",
      " ('D',\n",
      "  [TopLogprob(token='D', bytes=[68], logprob=-0.00017326632),\n",
      "   TopLogprob(token='A', bytes=[65], logprob=-9.179176),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-10.573698)]),\n",
      " ('C',\n",
      "  [TopLogprob(token='B', bytes=[66], logprob=-0.25176376),\n",
      "   TopLogprob(token='C', bytes=[67], logprob=-1.5055556),\n",
      "   TopLogprob(token='A', bytes=[65], logprob=-7.712375)]),\n",
      " ('B',\n",
      "  [TopLogprob(token='B', bytes=[66], logprob=-0.00011939728),\n",
      "   TopLogprob(token='A', bytes=[65], logprob=-10.099122),\n",
      "   TopLogprob(token=' B', bytes=[32, 66], logprob=-10.742089)]),\n",
      " ('C',\n",
      "  [TopLogprob(token='C', bytes=[67], logprob=-0.0016689957),\n",
      "   TopLogprob(token='B', bytes=[66], logprob=-6.4706535),\n",
      "   TopLogprob(token='A', bytes=[65], logprob=-9.932955)]),\n",
      " ('B',\n",
      "  [TopLogprob(token='B', bytes=[66], logprob=-0.00029947367),\n",
      "   TopLogprob(token='Choice', bytes=[67, 104, 111, 105, 99, 101], logprob=-9.208588),\n",
      "   TopLogprob(token='**', bytes=[42, 42], logprob=-9.336935)]),\n",
```

```
        " ('C',\n",
        "  [TopLogprob(token='C', bytes=[67], logprob=-0.004795596),\n",
        "   TopLogprob(token='B', bytes=[66], logprob=-5.376576),\n",
        "   TopLogprob(token='A', bytes=[65], logprob=-9.683446)]),\n",
        " ('B',\n",
        "  [TopLogprob(token='B', bytes=[66], logprob=-0.0016926733),\n",
        "   TopLogprob(token='C', bytes=[67], logprob=-6.4478045),\n",
        "   TopLogprob(token='Choice', bytes=[67, 104, 111, 105, 99, 101], logprob=-9.870015)]),\n",
        " ('A',\n",
        "  [TopLogprob(token='A', bytes=[65], logprob=-0.00012487332),\n",
        "   TopLogprob(token='B', bytes=[66], logprob=-9.616012),\n",
        "   TopLogprob(token=' A', bytes=[32, 65], logprob=-10.646184)]),\n",
        " ('C',\n",
        "  [TopLogprob(token='C', bytes=[67], logprob=-0.001123549),\n",
        "   TopLogprob(token='A', bytes=[65], logprob=-7.0805283),\n",
        "   TopLogprob(token='Choice', bytes=[67, 104, 111, 105, 99, 101], logprob=-8.686615)]),\n",
        " ('C',\n",
        "  [TopLogprob(token='C', bytes=[67], logprob=-0.0023396122),\n",
        "   TopLogprob(token='A', bytes=[65], logprob=-6.3256125),\n",
        "   TopLogprob(token='Is', bytes=[73, 115], logprob=-8.370381)]),\n",
        " ('B',\n",
        "  [TopLogprob(token='B', bytes=[66], logprob=-0.00017541199),\n",
        "   TopLogprob(token='**', bytes=[42, 42], logprob=-9.622995),\n",
        "   TopLogprob(token='Choice', bytes=[67, 104, 111, 105, 99, 101], logprob=-9.695049)]),\n",
        " ('B',\n",
        "  [TopLogprob(token='B', bytes=[66], logprob=-0.0010155413),\n",
        "   TopLogprob(token='C', bytes=[67], logprob=-6.9644785),\n",
        "   TopLogprob(token='A', bytes=[65], logprob=-10.900178)])]"
      ]
     },
     "execution_count": 10,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "run_experiments(base_prompts, results)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 11,
   "id": "549ee038-f949-4bd3-a8dd-4583109e5e05",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Correct cases are 19 out of 25 cases. Therefore, percentage is 76.0\n"
     ]
    }
   ],
   "source": [
    "def calculate_accuracy(results, df_pol):\n",
    "    correct = 0\n",
    "    for i in range(25):\n",
    "        actual_answer = df_pol.iloc[i]['answer']\n",
    "        predicted_answer = results[i][0]\n",
    "        if actual_answer == predicted_answer:\n",
    "            correct += 1\n",
    "    total = len(results)\n",
    "    accuracy = correct / total\n",
    "    print(f\"Correct cases are {correct} out of {total} cases. Therefore, percentage is {(correct/total)*100}\
    \n",
    "calculate_accuracy(results, df_pol)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "29af3e33-249f-4ab7-b614-00740079baa2",
   "metadata": {},
   "source": [
    "**Now let's do US Foreign with CoT.**\n",
    "\n",
    "The goal of this prompt is to provide few example cases with reasoning steps.\n",
    "We will provide 5 example cases (This can be changed)."
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "id": "716674b9-b996-410c-8bea-f21932653331",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
       "      <th></th>\n",
       "      <th>question</th>\n",
       "      <th>A</th>\n",
       "      <th>B</th>\n",
       "      <th>C</th>\n",
       "      <th>D</th>\n",
       "      <th>answer</th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>0</th>\n",
       "      <td>How did the 2008 financial crisis affect America's international reputation?</td>\n",
       "      <td>It damaged support for the US model of political economy and capitalism</td>\n",
       "      <td>It created anger at the United States for exaggerating the crisis</td>\n",
       "      <td>It increased support for American global leadership under President Obama</td>\n",
       "      <td>It reduced global use of the US dollar</td>\n",
       "      <td>A</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>1</th>\n",
       "      <td>How did NSC-68 change U.S. strategy?</td>\n",
       "      <td>It globalized containment.</td>\n",
       "      <td>It militarized containment.</td>\n",
       "      <td>It called for the development of the hydrogen bomb.</td>\n",
       "      <td>All of the above</td>\n",
       "      <td>D</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>2</th>\n",
       "      <td>The realm of policy decisions concerned primarily with relations between the United States and th
       "      <td>terrorism policy.</td>\n",
       "      <td>economic policy.</td>\n",
       "      <td>foreign policy.</td>\n",
       "      <td>international policy.</td>\n",
       "      <td>C</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>3</th>\n",
       "      <td>How do Defensive Realism and Offensive Realism differ in their explanation of state behaviour?</t
       "      <td>Defensive realists place greater emphasis on the role of international institutions</td>\n",
       "      <td>Defensive realists place less emphasis on geographical factors</td>\n",
       "      <td>Offensive realists give more priority to the national interest than Defensive realists.</td>\n",
       "      <td>Defensive realists believe states are security maximizers, while Offensive realists believe state
       "      <td>D</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>4</th>\n",
       "      <td>How did Donald Trump attack globalization in the 2016 campaign?</td>\n",
```

```
      "      <td>Globalization had made men like him too rich</td>\n",
      "      <td>Globalization only benefited certain American states, such as New York</td>\n",
      "      <td>Liberal elites had encouraged globalization, while 'ordinary Americans' lost jobs because of it</
      "      <td>Globalization encouraged damaging trade wars</td>\n",
      "      <td>C</td>\n",
      "    </tr>\n",
      "  </tbody>\n",
      "</table>\n",
      "</div>"
     ],
     "text/plain": [
      "
      "0                                                    How did the 2008 financial crisis affect America's i
      "1                                                                                          How did NSC-
      "2  The realm of policy decisions concerned primarily with relations between the United States and the rest
      "3                                  How do Defensive Realism and Offensive Realism differ in their explana
      "4                                                        How did Donald Trump attack globalizati
      "\n",
      "                                                                                        A  \\\n",
      "0             It damaged support for the US model of political economy and capitalism   \n",
      "1                                                    It globalized containment.   \n",
      "2                                                         terrorism policy.   \n",
      "3  Defensive realists place greater emphasis on the role of international institutions   \n",
      "4                                     Globalization had made men like him too rich   \n",
      "\n",
      "                                                                  B  \\\n",
      "0       It created anger at the United States for exaggerating the crisis   \n",
      "1                                          It militarized containment.   \n",
      "2                                                economic policy.   \n",
      "3          Defensive realists place less emphasis on geographical factors   \n",
      "4  Globalization only benefited certain American states, such as New York   \n",
      "\n",
      "                                                                                      C  \\\n",
      "0                      It increased support for American global leadership under President Obama   \n",
      "1                                       It called for the development of the hydrogen bomb.   \n",
      "2                                                                  foreign policy.   \n",
      "3          Offensive realists give more priority to the national interest than Defensive realists.   \n",
      "4  Liberal elites had encouraged globalization, while 'ordinary Americans' lost jobs because of it   \n",
      "\n",
      "                                                                                  D  \\\n",
      "0                                                                It reduced global use
      "1
      "2                                                                                inte
      "3  Defensive realists believe states are security maximizers, while Offensive realists believe states to be
      "4                                                                Globalization encouraged da
      "\n",
      "   answer  \n",
      "0      A  \n",
      "1      D  \n",
      "2      C  \n",
      "3      D  \n",
      "4      C  "
     ]
    },
    "execution_count": 12,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "#Generating CoT reasoning steps\n",
   "dev_pol = Path(r\"C:\\Users\\ASUS\\Desktop\\UW\\Winter 2024\\LING 575 D\\A6\\data\\data\\dev\\us_foreign_polic
   "df_dev_pol = pd.read_csv(dev_pol, names=['question', 'A', 'B', 'C', 'D', 'answer'])\n",
   "df_dev_pol.head()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 22,
  "id": "bc180c53-cfd8-4671-a73f-cf3e1468807d",
  "metadata": {},
  "outputs": [],
  "source": [
   "def make_zero_prompts(df):\n",
   "    \"\"\"\n",
   "    \"\"\"\n",
   "    prompts = []\n",
```

```
    "    system_instruct = {\"role\": \"system\", \"content\": \"Your task is to answer the following US-Foreign Po
    "    \n",
    "    for index, row in df.iterrows():\n",
    "        prompt_content = f\"{row['question']}\\nA: {row['A']}\\nB: {row['B']}\\nC: {row['C']}\\nD: {row['D']}\
    "        user_instruct = {\"role\": \"user\", \"content\": prompt_content}\n",
    "\n",
    "        prompt_list = [system_instruct, user_instruct]\n",
    "        prompts.append(prompt_list)\n",
    "    return prompts"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "id": "a953a85d-43e4-4aea-9e7e-fa2e5cb0fd3c",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "5\n",
      "5\n"
     ]
    }
   ],
   "source": [
    "zero_prompts = make_zero_prompts(df_dev_pol)\n",
    "print(len(zero_prompts))\n",
    "print(len(df_dev_pol))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "id": "ddfc2087-eb16-4077-990b-1d2ca8476d5d",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Let's optimize the function calls later\n",
    "def send_zero_prompt(MESSAGES):\n",
    "    MAX_NEW_TOKENS=150\n",
    "    \n",
    "    completion = client.chat.completions.create(\n",
    "          model='gpt-3.5-turbo',\n",
    "          messages=MESSAGES,\n",
    "          temperature=1.0,\n",
    "          max_tokens=MAX_NEW_TOKENS,\n",
    "          frequency_penalty=0.0,\n",
    "          presence_penalty=0.0\n",
    "        )\n",
    "\n",
    "    answer = completion.choices[0].message.content\n",
    "    return answer"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "id": "f28dc40f-3764-421a-8250-103024e48506",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "['A: It damaged support for the US model of political economy and capitalism',\n",
       " 'NSC-68 was a document produced by the United States National Security Council in 1950 during the Cold War',\n",
       " \"The correct answer is C: foreign policy.\\n\\nForeign policy is the realm of policy decisions concerned",\n",
       " 'Defensive Realism and Offensive Realism are two different theories within the realm of international rela',\n",
       " \"Step 1: Donald Trump's stance on globalization\\n- During his 2016 campaign, Donald Trump criticized glo"
      ]
     },
     "execution_count": 25,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
```

```
    "source": [
     "zero_results = []\n",
     "def run_zero_experiments(PROMPTS, results):\n",
     "    for prompt in PROMPTS:\n",
     "        # print(\"****\"*8)\n",
     "        # print()\n",
     "        \n",
     "        messages = prompt\n",
     "        # print(messages)\n",
     "        # print()\n",
     "        \n",
     "        cot_answer = send_zero_prompt(messages)\n",
     "        # print(f\"Model answer: {final_answer}\")\n",
     "        # log_probs_formatted = \", \".join([f\"{token.token}: {token.logprob:.4f}, {round(np.exp(token.logprc
     "        # print(f\"Log Probabilities: {log_probs_formatted}\")\n",
     "\n",
     "        zero_results.append(cot_answer)\n",
     "    return zero_results\n",
     "run_zero_experiments(zero_prompts, zero_results)"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 26,
    "id": "2906de3e-2a1d-497e-8a50-6c50f88c15e7",
    "metadata": {},
    "outputs": [
     {
      "data": {
       "text/plain": [
        "\"'A: It damaged support for the US model of political economy and capitalism'\"
       ]
      },
      "execution_count": 26,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "zero_results[0]"
    ]
   },
   {
    "cell_type": "markdown",
    "id": "6fc721cb-4e9a-4382-a15d-47acb66e335b",
    "metadata": {},
    "source": [
     "**Zero_results now contains example reasoning processes from machine. We will feed this into our previous prom
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 27,
    "id": "75c0f28e-f4ea-4af3-838a-275dd4f4884f",
    "metadata": {},
    "outputs": [],
    "source": [
     "def make_cot_prompts(df, df_dev, zero_results):\n",
     "    \"\"\"\n",
     "    \"\"\"\n",
     "    prompts = []\n",
     "    system_instruct = {\"role\": \"system\", \"content\": \"Your task is to answer the following US-Foreign Pc
     "\n",
     "    # Chain of thought prompts\n",
     "    # user_instruct: each entry of df_dev similar to promt\n",
     "    # assisnt_assistant_response: each entry of zero_results\n",
     "    cot_steps = []\n",
     "    for index, row in df_dev.iterrows():\n",
     "        question = f\"{row['question']}\\nA: {row['A']}\\nB: {row['B']}\\nC: {row['C']}\\nD: {row['D']}\"\n",
     "        explanation = zero_results[index]\n",
     "        cot_steps.append({\"role\": \"user\", \"content\": question})\n",
     "        cot_steps.append({\"role\": \"assistant\", \"content\": explanation})\n",
     "            \n",
     "    for index, row in df.iterrows():\n",
     "        prompt_content = f\"{row['question']}\\nA: {row['A']}\\nB: {row['B']}\\nC: {row['C']}\\nD: {row['D']}\
     "        final_question = {\"role\": \"user\", \"content\": prompt_content}\n",
     "\n",
```

```
    "        prompt_list = [system_instruct] + cot_steps + [final_question]\n",
    "        prompts.append(prompt_list)\n",
    "    return prompts"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 28,
   "id": "9db93fa2-e9e9-4b4c-acec-2e4ffee22f9d",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "[{'role': 'system',\n",
       "  'content': 'Your task is to answer the following US-Foreign Policy question by picking the correct answer
       " {'role': 'user',\n",
       "  'content': \"How did the 2008 financial crisis affect America's international reputation?\\nA: It damaged
       " {'role': 'assistant',\n",
       "  'content': 'A: It damaged support for the US model of political economy and capitalism'},\n",
       " {'role': 'user',\n",
       "  'content': 'How did NSC-68 change U.S. strategy?\\nA: It globalized containment.\\nB: It militarized cont
       " {'role': 'assistant',\n",
       "  'content': 'NSC-68 was a document produced by the United States National Security Council in 1950 during
       " {'role': 'user',\n",
       "  'content': 'The realm of policy decisions concerned primarily with relations between the United States ar
       " {'role': 'assistant',\n",
       "  'content': \"The correct answer is C: foreign policy.\\n\\nForeign policy is the realm of policy decisior
       " {'role': 'user',\n",
       "  'content': 'How do Defensive Realism and Offensive Realism differ in their explanation of state behaviour
       " {'role': 'assistant',\n",
       "  'content': 'Defensive Realism and Offensive Realism are two different theories within the realm of interr
       " {'role': 'user',\n",
       "  'content': \"How did Donald Trump attack globalization in the 2016 campaign?\\nA: Globalization had made
       " {'role': 'assistant',\n",
       "  'content': \"Step 1: Donald Trump's stance on globalization\\n- During his 2016 campaign, Donald Trump cr
       " {'role': 'user',\n",
       "  'content': 'What is the structure of the United Nations Security Council?\\nA: 5 permanent members with v
      ]
     },
     "execution_count": 28,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "cot_prompts = make_cot_prompts(df_pol, df_dev_pol, zero_results)\n",
    "cot_prompts[0]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 29,
   "id": "c6fc4c29-a08d-4cdb-9fe7-ebd3900a3d75",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "[('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0016088859),\n",
       "   TopLogprob(token='B', bytes=[66], logprob=-7.6235895),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.990781)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0004156569),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-8.304558),\n",
       "   TopLogprob(token='\\n', bytes=[10], logprob=-10.401033)]),\n",
       " ('D',\n",
       "  [TopLogprob(token='D', bytes=[68], logprob=-0.0024349974),\n",
       "   TopLogprob(token='China', bytes=[67, 104, 105, 110, 97], logprob=-6.835405),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-8.107663)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0010962842),\n",
       "   TopLogprob(token='Richard', bytes=[82, 105, 99, 104, 97, 114, 100], logprob=-7.4031215),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.923858)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.00040111772),\n",
```

```
"   TopLogprob(token=' A', bytes=[32, 65], logprob=-8.43971),\n",
"   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-10.293017)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.047426958),\n",
"   TopLogprob(token='C', bytes=[67], logprob=-4.69048),\n",
"   TopLogprob(token='Imp', bytes=[73, 109, 112], logprob=-4.758235)]),\n",
" ('A',\n",
"  [TopLogprob(token='A', bytes=[65], logprob=-0.017273812),\n",
"   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-4.939452),\n",
"   TopLogprob(token=' A', bytes=[32, 65], logprob=-6.181794)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.047406383),\n",
"   TopLogprob(token='A', bytes=[65], logprob=-3.7697995),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-4.7622232)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.00092885265),\n",
"   TopLogprob(token=' D', bytes=[32, 68], logprob=-7.7651825),\n",
"   TopLogprob(token='World', bytes=[87, 111, 114, 108, 100], logprob=-9.338926)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.015263107),\n",
"   TopLogprob(token='In', bytes=[73, 110], logprob=-4.512217),\n",
"   TopLogprob(token='the', bytes=[116, 104, 101], logprob=-5.5421243)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.00878498),\n",
"   TopLogprob(token='How', bytes=[72, 111, 119], logprob=-4.926967),\n",
"   TopLogprob(token=' B', bytes=[32, 66], logprob=-7.1120124)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.5183536),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-1.1768581),\n",
"   TopLogprob(token='C', bytes=[67], logprob=-2.9464827)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0051979437),\n",
"   TopLogprob(token='According', bytes=[65, 99, 99, 111, 114, 100, 105, 110, 103], logprob=-5.598515),\n",
"   TopLogprob(token='System', bytes=[83, 121, 115, 116, 101, 109], logprob=-7.3928285)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.004278077),\n",
"   TopLogprob(token='All', bytes=[65, 108, 108], logprob=-6.8340454),\n",
"   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-7.2798176)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.03885733),\n",
"   TopLogprob(token='Within', bytes=[87, 105, 116, 104, 105, 110], logprob=-3.3736255),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-6.0991354)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.018950699),\n",
"   TopLogprob(token='What', bytes=[87, 104, 97, 116], logprob=-4.361171),\n",
"   TopLogprob(token='Mos', bytes=[77, 111, 115], logprob=-6.4463663)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.011508117),\n",
"   TopLogprob(token='What', bytes=[87, 104, 97, 116], logprob=-5.4051714),\n",
"   TopLogprob(token='An', bytes=[65, 110], logprob=-5.504655)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0011102092),\n",
"   TopLogprob(token=' B', bytes=[32, 66], logprob=-7.8516273),\n",
"   TopLogprob(token='International', bytes=[73, 110, 116, 101, 114, 110, 97, 116, 105, 111, 110, 97, 108],
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.0054827603),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-6.504607),\n",
"   TopLogprob(token='After', bytes=[65, 102, 116, 101, 114], logprob=-7.1022024)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.008505956),\n",
"   TopLogprob(token='Lib', bytes=[76, 105, 98], logprob=-5.692668),\n",
"   TopLogprob(token='Why', bytes=[87, 104, 121], logprob=-5.7236013)]),\n",
" ('A',\n",
"  [TopLogprob(token='A', bytes=[65], logprob=-0.0015402117),\n",
"   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.2433825),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-8.316039)]),\n",
" ('D',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.39183927),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-1.2414505),\n",
"   TopLogprob(token='It', bytes=[73, 116], logprob=-4.474206)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.001592346),\n",
"   TopLogprob(token=' C', bytes=[32, 67], logprob=-7.1249995),\n",
"   TopLogprob(token='Is', bytes=[73, 115], logprob=-8.554554)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0011824887),\n",
"   TopLogprob(token=' B', bytes=[32, 66], logprob=-8.046352),\n",
```

```
        "   TopLogprob(token='To', bytes=[84, 111], logprob=-8.087257)]),\n",
        " ('B',\n",
        "  [TopLogprob(token='B', bytes=[66], logprob=-0.0025892158),\n",
        "   TopLogprob(token='George', bytes=[71, 101, 111, 114, 103, 101], logprob=-6.832988),\n",
        "   TopLogprob(token='C', bytes=[67], logprob=-7.4979086)])]"
       ]
      },
      "execution_count": 29,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "cot_results = []\n",
     "run_experiments(cot_prompts, cot_results)"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 30,
    "id": "70f71ade-bc2f-4fec-8ddd-1382b3c0c168",
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Correct cases are 18 out of 25 cases. Therefore, percentage is 72.0\n"
      ]
     }
    ],
    "source": [
     "calculate_accuracy(cot_results, df_pol)"
    ]
   },
   {
    "cell_type": "markdown",
    "id": "f50ae3d3-60d1-4372-a367-7cebc1a4dcfd",
    "metadata": {},
    "source": [
     "**CoT examples did not result in any changes with 25 instnaces, second run resulted in one more wrong**\n",
     "\n",
     "Now we are going to test corruptions: let's start by changing one verb "
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 33,
    "id": "9131c7f8-ad2f-423e-a3ea-71a1d945b259",
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Requirement already satisfied: nltk in c:\\users\\asus\\anaconda3\\envs\\wrench\\lib\\site-packages (3.8.1)\
       "Requirement already satisfied: click in c:\\users\\asus\\anaconda3\\envs\\wrench\\lib\\site-packages (from r
       "Requirement already satisfied: joblib in c:\\users\\asus\\anaconda3\\envs\\wrench\\lib\\site-packages (from
       "Requirement already satisfied: regex>=2021.8.3 in c:\\users\\asus\\anaconda3\\envs\\wrench\\lib\\site-packag
       "Requirement already satisfied: tqdm in c:\\users\\asus\\anaconda3\\envs\\wrench\\lib\\site-packages (from nl
       "Requirement already satisfied: colorama in c:\\users\\asus\\anaconda3\\envs\\wrench\\lib\\site-packages (fro
      ]
     }
    ],
    "source": [
     "# if you don't have these please download them\n",
     "!pip install nltk"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 34,
    "id": "bfc97119-4ea8-4f63-8230-c4c4d7e4605b",
    "metadata": {},
    "outputs": [],
    "source": [
     "# nltk.download('averaged_perceptron_tagger')\n",
```

```
    "# nltk.download('wordnet')\n",
    "# nltk.download('omw-1.4')\n",
    "# nltk.download('punkt')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 35,
   "id": "f35d2c52-95db-4ab0-a5f8-e40503839768",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "changed damaged into not damaged\n",
      "changed choosing into not choosing\n",
      "changed made into unmake\n",
      "changed do into unmake\n",
      "changed Analyzing into synthesize\n",
      "A: It damaged support for the US model of political economy and capitalism\n",
      "A: It not damaged support for the US model of political economy and capitalism\n"
     ]
    }
   ],
   "source": [
    "# changing one verb from the cot_steps\n",
    "import nltk\n",
    "from nltk import pos_tag, word_tokenize  # Corrected import statement\n",
    "from nltk.corpus import wordnet as wn\n",
    "from random import choice\n",
    "\n",
    "def get_antonym(word):\n",
    "    \"\"\"Find an antonym for the word.\"\"\"\n",
    "    antonyms = []\n",
    "    for syn in wn.synsets(word, pos=wn.VERB):\n",
    "        for lemma in syn.lemmas():\n",
    "            if lemma.antonyms():\n",
    "                antonyms.append(lemma.antonyms()[0].name())\n",
    "    return choice(antonyms) if antonyms else None\n",
    "\n",
    "def replace_verb_with_antonym(text):\n",
    "    \"\"\"Replace a randomly selected verb with its antonym.\"\"\"\n",
    "    tokens = word_tokenize(text)\n",
    "    tagged_tokens = pos_tag(tokens)\n",
    "    \n",
    "    # Identify all verbs\n",
    "    verbs = [word for word, tag in tagged_tokens if tag.startswith('VB')]\n",
    "    \n",
    "    if verbs:\n",
    "        # Randomly select a verb\n",
    "        selected_verb = choice(verbs)\n",
    "        \n",
    "        # Find an antonym\n",
    "        antonym = get_antonym(selected_verb)\n",
    "        \n",
    "        if antonym:\n",
    "            # Replace the first occurrence of the verb with its antonym\n",
    "            text = text.replace(selected_verb, antonym, 1)\n",
    "        else:\n",
    "            antonym = f\"not {selected_verb}\"\n",
    "            text = text.replace(selected_verb, antonym, 1)\n",
    "    print(f\"changed {selected_verb} into {antonym}\")\n",
    "    return text\n",
    "\n",
    "def process_zero_results(zero_results):\n",
    "    \"\"\"Process each entry in zero_results to replace a verb with its antonym.\"\"\"\n",
    "    updated_results = []\n",
    "    for entry in zero_results:\n",
    "        updated_entry = replace_verb_with_antonym(entry)\n",
    "        updated_results.append(updated_entry)\n",
    "    return updated_results\n",
    "\n",
    "cot_verb_corrupted = process_zero_results(zero_results)\n",
    "print(zero_results[0])\n",
    "print(cot_verb_corrupted[0])"
   ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": 36,
   "id": "00e637ba-82a2-4b8a-a984-ae85031a8564",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "[{'role': 'system',\n",
       "  'content': 'Your task is to answer the following US-Foreign Policy question by picking the correct answer
       " {'role': 'user',\n",
       "  'content': \"How did the 2008 financial crisis affect America's international reputation?\\nA: It damagec
       " {'role': 'assistant',\n",
       "  'content': 'A: It not damaged support for the US model of political economy and capitalism'},\n",
       " {'role': 'user',\n",
       "  'content': 'How did NSC-68 change U.S. strategy?\\nA: It globalized containment.\\nB: It militarized cont
       " {'role': 'assistant',\n",
       "  'content': 'NSC-68 was a document produced by the United States National Security Council in 1950 during
       " {'role': 'user',\n",
       "  'content': 'The realm of policy decisions concerned primarily with relations between the United States ar
       " {'role': 'assistant',\n",
       "  'content': \"The correct answer is C: foreign policy.\\n\\nForeign policy is the realm of policy decisior
       " {'role': 'user',\n",
       "  'content': 'How do Defensive Realism and Offensive Realism differ in their explanation of state behaviour
       " {'role': 'assistant',\n",
       "  'content': 'Defensive Realism and Offensive Realism are two different theories within the realm of interr
       " {'role': 'user',\n",
       "  'content': \"How did Donald Trump attack globalization in the 2016 campaign?\\nA: Globalization had made
       " {'role': 'assistant',\n",
       "  'content': \"Step 1: Donald Trump's stance on globalization\\n- During his 2016 campaign, Donald Trump cr
       " {'role': 'user',\n",
       "  'content': 'What is the structure of the United Nations Security Council?\\nA: 5 permanent members with v
      ]
     },
     "execution_count": 36,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "# Testing with corrupted outputs:\n",
    "cot_verb_corrupt_prompts = make_cot_prompts(df_pol, df_dev_pol, cot_verb_corrupted)\n",
    "cot_verb_corrupt_prompts[0]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "id": "648e84cf-ed20-493f-bd84-1a682d9853c6",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "[('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.040428642),\n",
       "   TopLogprob(token='B', bytes=[66], logprob=-3.551167),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-6.218701)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0004586711),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-8.212396),\n",
       "   TopLogprob(token='\\n', bytes=[10], logprob=-9.988107)]),\n",
       " ('D',\n",
       "  [TopLogprob(token='D', bytes=[68], logprob=-0.007311165),\n",
       "   TopLogprob(token='China', bytes=[67, 104, 105, 110, 97], logprob=-5.91837),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-6.564221)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0010864014),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.532302),\n",
       "   TopLogprob(token='Richard', bytes=[82, 105, 99, 104, 97, 114, 100], logprob=-7.770112)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.00046117438),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-8.681707),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-9.729139)]),\n",
       " ('D',\n",
       "  [TopLogprob(token='D', bytes=[68], logprob=-0.00046117438),\n",
       "   TopLogprob(token=' D', bytes=[32, 68], logprob=-8.681707),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-9.729139)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0010864014),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.532302),\n",
       "   TopLogprob(token='Richard', bytes=[82, 105, 99, 104, 97, 114, 100], logprob=-7.770112)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0004586711),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-8.212396),\n",
       "   TopLogprob(token='\\n', bytes=[10], logprob=-9.988107)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.040428642),\n",
       "   TopLogprob(token='B', bytes=[66], logprob=-3.551167),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-6.218701)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0004586711),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-8.212396),\n",
       "   TopLogprob(token='\\n', bytes=[10], logprob=-9.988107)]),\n",
       " ('D',\n",
       "  [TopLogprob(token='D', bytes=[68], logprob=-0.007311165),\n",
       "   TopLogprob(token='China', bytes=[67, 104, 105, 110, 97], logprob=-5.91837),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-6.564221)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0010864014),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.532302),\n",
       "   TopLogprob(token='Richard', bytes=[82, 105, 99, 104, 97, 114, 100], logprob=-7.770112)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.00046117438),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-8.681707),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-9.729139)]),\n",
       " ('D',\n",
       "  [TopLogprob(token='D', bytes=[68], logprob=-0.00046117438),\n",
       "   TopLogprob(token=' D', bytes=[32, 68], logprob=-8.681707),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-9.729139)])]
```

```
"  [TopLogprob(token='D', bytes=[68], logprob=-0.07057171),\n",
"   TopLogprob(token='C', bytes=[67], logprob=-3.5794446),\n",
"   TopLogprob(token='American', bytes=[65, 109, 101, 114, 105, 99, 97, 110], logprob=-5.161276)]),\n",
" ('A',\n",
"  [TopLogprob(token='A', bytes=[65], logprob=-0.081883274),\n",
"   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-4.042369),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-4.5211554)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.2559022),\n",
"   TopLogprob(token='A', bytes=[65], logprob=-1.9870957),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-2.680482)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.0011459336),\n",
"   TopLogprob(token=' D', bytes=[32, 68], logprob=-7.695297),\n",
"   TopLogprob(token='How', bytes=[72, 111, 119], logprob=-8.946722)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.030814292),\n",
"   TopLogprob(token='In', bytes=[73, 110], logprob=-3.8030922),\n",
"   TopLogprob(token='the', bytes=[116, 104, 101], logprob=-4.8884554)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.00319209),\n",
"   TopLogprob(token='How', bytes=[72, 111, 119], logprob=-6.066472),\n",
"   TopLogprob(token=' B', bytes=[32, 66], logprob=-7.662465)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.18254574),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-2.403287),\n",
"   TopLogprob(token='C', bytes=[67], logprob=-3.2942266)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0077449214),\n",
"   TopLogprob(token='According', bytes=[65, 99, 99, 111, 114, 100, 105, 110, 103], logprob=-5.439296),\n",
"   TopLogprob(token='System', bytes=[83, 121, 115, 116, 101, 109], logprob=-6.651902)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.011766402),\n",
"   TopLogprob(token='All', bytes=[65, 108, 108], logprob=-5.8203373),\n",
"   TopLogprob(token='ALL', bytes=[65, 76, 76], logprob=-6.3907013)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.061507974),\n",
"   TopLogprob(token='Within', bytes=[87, 105, 116, 104, 105, 110], logprob=-2.9824102),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-5.076935)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.006690749),\n",
"   TopLogprob(token='What', bytes=[87, 104, 97, 116], logprob=-5.7615814),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-6.927696)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.0060399505),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-6.30833),\n",
"   TopLogprob(token='What', bytes=[87, 104, 97, 116], logprob=-6.5819564)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0012764268),\n",
"   TopLogprob(token=' B', bytes=[32, 66], logprob=-7.9254594),\n",
"   TopLogprob(token='International', bytes=[73, 110, 116, 101, 114, 110, 97, 116, 105, 111, 110, 97, 108],
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.0028006025),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-7.396462),\n",
"   TopLogprob(token=' C', bytes=[32, 67], logprob=-7.4207025)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.023480257),\n",
"   TopLogprob(token='Lib', bytes=[76, 105, 98], logprob=-4.13696),\n",
"   TopLogprob(token='Why', bytes=[87, 104, 121], logprob=-5.3894777)]),\n",
" ('A',\n",
"  [TopLogprob(token='A', bytes=[65], logprob=-0.0023098846),\n",
"   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.1690845),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-7.8429623)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.26469994),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-1.7243729),\n",
"   TopLogprob(token='It', bytes=[73, 116], logprob=-3.6092167)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.0023947945),\n",
"   TopLogprob(token=' C', bytes=[32, 67], logprob=-7.290702),\n",
"   TopLogprob(token='Is', bytes=[73, 115], logprob=-7.7688417)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0019228025),\n",
"   TopLogprob(token='What', bytes=[87, 104, 97, 116], logprob=-7.0502243),\n",
"   TopLogprob(token=' B', bytes=[32, 66], logprob=-8.192373)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0034640562),\n",
```

```
        "   TopLogprob(token='George', bytes=[71, 101, 111, 114, 103, 101], logprob=-6.762034),\n",
        "   TopLogprob(token='C', bytes=[67], logprob=-7.426575)])]"
       ]
      },
      "execution_count": 37,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "cot_verb_corrupt_results = []\n",
     "run_experiments(cot_verb_corrupt_prompts, cot_verb_corrupt_results)"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 38,
    "id": "595f4f92-4c75-4a5a-bb73-d096b9a7a4b0",
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Correct cases are 19 out of 25 cases. Therefore, percentage is 76.0\n"
      ]
     }
    ],
    "source": [
     "calculate_accuracy(cot_verb_corrupt_results, df_pol)"
    ]
   },
   {
    "cell_type": "markdown",
    "id": "f0abcac6-ffe7-4ac3-97d9-56ba1c37c19e",
    "metadata": {},
    "source": [
     "***So the corrupting one verb did not change anything as well***\n",
     "\n",
     "Now let's just add 'not' to every verbs"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 39,
    "id": "70b59d44-ef06-4661-b291-7362e7183362",
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "NSC-68 not was a document not produced by the United States National Security Council in 1950 during the Col
      ]
     }
    ],
    "source": [
     "def negate_all_verbs(text):\n",
     "    \"\"\"\n",
     "    Negate all verbs in the text by adding 'not' before each verb.\n",
     "    \"\"\"\n",
     "    tokens = word_tokenize(text)\n",
     "    tagged_tokens = pos_tag(tokens)\n",
     "    \n",
     "    # Process tokens in reverse order to avoid indexing issues when inserting 'not'\n",
     "    for i in reversed(range(len(tagged_tokens))):\n",
     "        word, tag = tagged_tokens[i]\n",
     "        if tag.startswith('VB'):  # If the token is a verb\n",
     "            tokens.insert(i, 'not')\n",
     "    \n",
     "    return ' '.join(tokens)\n",
     "\n",
     "def process_zero_results(zero_results):\n",
     "    \"\"\"\n",
     "    Process each entry in zero_results to negate all verbs by adding 'not'.\n",
     "    \"\"\"\n",
     "    updated_results = []\n",
```

```
    "    for entry in zero_results:\n",
    "        updated_entry = negate_all_verbs(entry)\n",
    "        updated_results.append(updated_entry)\n",
    "    return updated_results\n",
    "\n",
    "cot_negative_verb = process_zero_results(zero_results)\n",
    "print(cot_negative_verb[1])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 40,
   "id": "18089f87-7031-4273-9071-fb5f1d65ce50",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "[{'role': 'system',\n",
       "  'content': 'Your task is to answer the following US-Foreign Policy question by picking the correct answer
       " {'role': 'user',\n",
       "  'content': \"How did the 2008 financial crisis affect America's international reputation?\\nA: It damaged
       " {'role': 'assistant',\n",
       "  'content': 'A : It not damaged support for the US model of political economy and capitalism'},\n",
       " {'role': 'user',\n",
       "  'content': 'How did NSC-68 change U.S. strategy?\\nA: It globalized containment.\\nB: It militarized cont
       " {'role': 'assistant',\n",
       "  'content': 'NSC-68 not was a document not produced by the United States National Security Council in 1950
       " {'role': 'user',\n",
       "  'content': 'The realm of policy decisions concerned primarily with relations between the United States ar
       " {'role': 'assistant',\n",
       "  'content': \"The correct answer not is C : foreign policy . Foreign policy not is the realm of policy dec
       " {'role': 'user',\n",
       "  'content': 'How do Defensive Realism and Offensive Realism differ in their explanation of state behaviour
       " {'role': 'assistant',\n",
       "  'content': 'Defensive Realism and Offensive Realism not are two different theories within the realm of ir
       " {'role': 'user',\n",
       "  'content': \"How did Donald Trump attack globalization in the 2016 campaign?\\nA: Globalization had made
       " {'role': 'assistant',\n",
       "  'content': \"Step 1 : Donald Trump 's stance on globalization - During his 2016 campaign , Donald Trump r
       " {'role': 'user',\n",
       "  'content': 'What is the structure of the United Nations Security Council?\\nA: 5 permanent members with v
      ]
     },
     "execution_count": 40,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "cot_negative_verb_corrupt_prompts = make_cot_prompts(df_pol, df_dev_pol, cot_negative_verb)\n",
    "cot_negative_verb_corrupt_prompts[0]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 41,
   "id": "21a587c3-e724-4357-9621-8559ac8b6485",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "[('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.08668797),\n",
       "   TopLogprob(token='B', bytes=[66], logprob=-2.6347742),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-6.1522627)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.001406544),\n",
       "   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.2106085),\n",
       "   TopLogprob(token='It', bytes=[73, 116], logprob=-9.3435755)]),\n",
       " ('D',\n",
       "  [TopLogprob(token='D', bytes=[68], logprob=-0.0037107659),\n",
       "   TopLogprob(token='China', bytes=[67, 104, 105, 110, 97], logprob=-6.2810926),\n",
       "   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-8.065318)]),\n",
       " ('A',\n",
       "  [TopLogprob(token='A', bytes=[65], logprob=-0.0026580587),\n",
```

```
"   TopLogprob(token='Richard', bytes=[82, 105, 99, 104, 97, 114, 100], logprob=-6.7196302),\n",
"   TopLogprob(token=' A', bytes=[32, 65], logprob=-6.870858)]),\n",
" ('A',\n",
"  [TopLogprob(token='A', bytes=[65], logprob=-0.0020480782),\n",
"   TopLogprob(token=' A', bytes=[32, 65], logprob=-7.5182815),\n",
"   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-7.9600463)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.023657355),\n",
"   TopLogprob(token='C', bytes=[67], logprob=-4.272448),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-6.1036267)]),\n",
" ('A',\n",
"  [TopLogprob(token='A', bytes=[65], logprob=-0.091935165),\n",
"   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-3.7246294),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-4.2425547)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.27038687),\n",
"   TopLogprob(token='A', bytes=[65], logprob=-1.9095318),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-2.6192238)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.0014808172),\n",
"   TopLogprob(token=' D', bytes=[32, 68], logprob=-7.6201444),\n",
"   TopLogprob(token='**', bytes=[42, 42], logprob=-8.825425)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.005062264),\n",
"   TopLogprob(token='the', bytes=[116, 104, 101], logprob=-6.163374),\n",
"   TopLogprob(token='In', bytes=[73, 110], logprob=-6.1983166)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0012840448),\n",
"   TopLogprob(token=' B', bytes=[32, 66], logprob=-7.5409055),\n",
"   TopLogprob(token='.', bytes=[46], logprob=-8.526825)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.15629102),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-2.4027262),\n",
"   TopLogprob(token='C', bytes=[67], logprob=-3.6426077)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0072212266),\n",
"   TopLogprob(token='According', bytes=[65, 99, 99, 111, 114, 100, 105, 110, 103], logprob=-5.5303917),\n",
"   TopLogprob(token='System', bytes=[83, 121, 115, 116, 101, 109], logprob=-6.505846)]),\n",
" ('D',\n",
"  [TopLogprob(token='D', bytes=[68], logprob=-0.005767142),\n",
"   TopLogprob(token='All', bytes=[65, 108, 108], logprob=-6.907676),\n",
"   TopLogprob(token='ALL', bytes=[65, 76, 76], logprob=-6.9668803)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.10003033),\n",
"   TopLogprob(token='Within', bytes=[87, 105, 116, 104, 105, 110], logprob=-3.0390506),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-3.1889567)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.015832579),\n",
"   TopLogprob(token='Mos', bytes=[77, 111, 115], logprob=-5.7075496),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-5.719045)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.010843711),\n",
"   TopLogprob(token='An', bytes=[65, 110], logprob=-5.313605),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-6.044331)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0015690223),\n",
"   TopLogprob(token=' B', bytes=[32, 66], logprob=-7.1216974),\n",
"   TopLogprob(token='**', bytes=[42, 42], logprob=-8.672437)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.0050220555),\n",
"   TopLogprob(token='B', bytes=[66], logprob=-6.367155),\n",
"   TopLogprob(token=' C', bytes=[32, 67], logprob=-6.607935)]),\n",
" ('B',\n",
"  [TopLogprob(token='B', bytes=[66], logprob=-0.0031956548),\n",
"   TopLogprob(token='Lib', bytes=[76, 105, 98], logprob=-7.3548403),\n",
"   TopLogprob(token='Option', bytes=[79, 112, 116, 105, 111, 110], logprob=-7.6268835)]),\n",
" ('A',\n",
"  [TopLogprob(token='A', bytes=[65], logprob=-0.00511268),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-6.309518),\n",
"   TopLogprob(token=' A', bytes=[32, 65], logprob=-6.575013)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.2659652),\n",
"   TopLogprob(token='D', bytes=[68], logprob=-1.6411216),\n",
"   TopLogprob(token='Oil', bytes=[79, 105, 108], logprob=-4.464516)]),\n",
" ('C',\n",
"  [TopLogprob(token='C', bytes=[67], logprob=-0.0046926113),\n",
"   TopLogprob(token=' C', bytes=[32, 67], logprob=-6.35368),\n",
```

```
        "   TopLogprob(token='Is', bytes=[73, 115], logprob=-6.775307)]),\n",
        " ('B',\n",
        "  [TopLogprob(token='B', bytes=[66], logprob=-0.0027377089),\n",
        "   TopLogprob(token=' B', bytes=[32, 66], logprob=-7.262917),\n",
        "   TopLogprob(token='.', bytes=[46], logprob=-7.937985)]),\n",
        " ('B',\n",
        "  [TopLogprob(token='B', bytes=[66], logprob=-0.0023990783),\n",
        "   TopLogprob(token='George', bytes=[71, 101, 111, 114, 103, 101], logprob=-7.1466517),\n",
        "   TopLogprob(token=' B', bytes=[32, 66], logprob=-7.6890655)])]"
       ]
      },
      "execution_count": 41,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "cot_negative_verb_corrupt_results = []\n",
     "run_experiments(cot_negative_verb_corrupt_prompts, cot_negative_verb_corrupt_results)"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 42,
    "id": "f0c45811-09b5-439a-b565-2ddb4f508fef",
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Correct cases are 19 out of 25 cases. Therefore, percentage is 76.0\n"
      ]
     }
    ],
    "source": [
     "calculate_accuracy(cot_negative_verb_corrupt_results, df_pol)"
    ]
   }
  ],
  "metadata": {
   "kernelspec": {
    "display_name": "Python 3 (ipykernel)",
    "language": "python",
    "name": "python3"
   },
   "language_info": {
    "codemirror_mode": {
     "name": "ipython",
     "version": 3
    },
    "file_extension": ".py",
    "mimetype": "text/x-python",
    "name": "python",
    "nbconvert_exporter": "python",
    "pygments_lexer": "ipython3",
    "version": "3.8.18"
   }
  },
  "nbformat": 4,
  "nbformat_minor": 5
}
```