



## THEORY OF COMPUTATION (CSE2002)

NAME	ANITEJ SRIVASTAVA
REGISTRATION NUMBER	19BCE0835
SEMESTER	WINSEM 20-21
COURSE	CSE2002
FACULTY	Dr. Viswanathan P
SLOT	A2
ASSIGNMENT NUMBER	1
NO. OF MEMBERS	1

### GOOGLE COLAB FILE LINK:

<https://colab.research.google.com/drive/1hIkZl6yTsHPdA3l-xDo8ZquDjpEoePD8?usp=sharing>

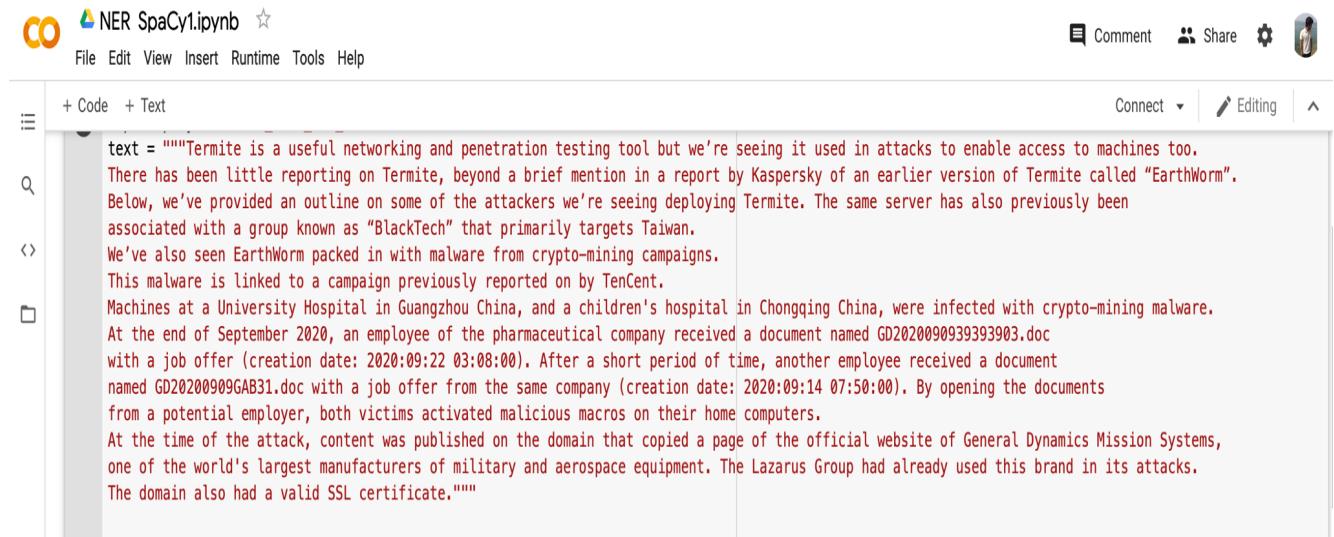
**Directed Acyclic Graph Construction for Malware Detection/Analysis in Cybersecurity & Digital Forensics Domain**

## Step 1: Acquiring raw text with respect to malware attacks & threats in the cybersecurity domain from AlienVault - Open Threat Exchange

The screenshot shows the AlienVault OTX Database interface. At the top, there is a navigation bar with links for Dashboard, Browse, Scan Endpoints, Create Pulse, Submit Sample, API Integration, and a search bar. Below the search bar, there are filters for Pulses (671), Users (0), Groups (10), Indicators (0), Malware Families (0), Industries (0), and Adversaries (1). A dropdown menu shows 'All' selected under 'cyber security'. The main content area displays search results for "cyber security" with 682 results found. The first result is titled "Lazarus Group Recruitment: Threat Hunters vs Head Hunters" and includes a green alien icon, a creation date of 2 days ago, and a link to the full entry. The second result is titled "SolarWinds: Advancing the Story" and includes a green alien icon, a creation date of 4 days ago, and a link to the full entry. The third result is titled "CISA Identifies SUPERNOVA Malware During Incident Response" and includes a green alien icon, a creation date of 5 days ago, and a link to the full entry. At the bottom of the page is a toolbar with various icons for sharing and interacting with the content.

Figure 1: AlienVault - OTX Database

Collected the data by scraping text from all the pulse reference links on the OTX platform.



The screenshot shows a Google Colaboratory notebook titled "NER SpaCy1.ipynb". The code cell contains the following Python code:

```
text = """Termite is a useful networking and penetration testing tool but we're seeing it used in attacks to enable access to machines too. There has been little reporting on Termite, beyond a brief mention in a report by Kaspersky of an earlier version of Termite called "EarthWorm". Below, we've provided an outline on some of the attackers we're seeing deploying Termite. The same server has also previously been associated with a group known as "BlackTech" that primarily targets Taiwan. We've also seen EarthWorm packed in with malware from crypto-mining campaigns. This malware is linked to a campaign previously reported on by TenCent. Machines at a University Hospital in Guangzhou China, and a children's hospital in Chongqing China, were infected with crypto-mining malware. At the end of September 2020, an employee of the pharmaceutical company received a document named GD20200909393903.doc with a job offer (creation date: 2020:09:22 03:08:00). After a short period of time, another employee received a document named GD20200909GAB31.doc with a job offer from the same company (creation date: 2020:09:14 07:50:00). By opening the documents from a potential employer, both victims activated malicious macros on their home computers. At the time of the attack, content was published on the domain that copied a page of the official website of General Dynamics Mission Systems, one of the world's largest manufacturers of military and aerospace equipment. The Lazarus Group had already used this brand in its attacks. The domain also had a valid SSL certificate."""
```

Figure 2: Text compiled from OTX Pulses

**Step 2:** Performing Named Entity Recognition (NER) on the raw text either using the built in labels or using custom annotations on the raw text:-

**i) With built in labels for entities** (without custom annotations on text)

I am using **spaCy** with Python on Google Colaboratory to perform Named Entity Recognition on the text acquired in step 1. SpaCy is an open-source software library for advanced natural language processing, written in the programming languages Python.

The “en\_core\_web\_sm” is a trained English pipeline optimized for CPU.

```

#ANITEJ SRIVASTAVA
#19BCE0835
import pandas as pd
import numpy as np
import spacy
from spacy import displacy
spacy.__version__
'2.2.4'

[3] nlp = spacy.load("en_core_web_sm")
text = """Termite is a useful networking and penetration testing tool but we're seeing it used in attacks to enable access to machines too. There has been little reporting on Termite, beyond a brief mention in a report by Kaspersky of an earlier version of Termite called EarthWorm. Below, we've provided an outline on some of the attackers we're seeing deploying Termite. The same server has also previously been associated with a group known as BlackTech that primarily targets Taiwan. We've also seen EarthWorm packed in with malware from crypto-mining campaigns. This malware is linked to a campaign previously reported on by TenCent. Machines at a University Hospital in Guangzhou China, and a children's hospital in Chongqing China, were infected with crypto-mining malware. At the end of September 2020, an employee of the pharmaceutical company received a document named GD20200909393903.doc with a job offer (creation date: 2020:09:22 03:08:00). After a short period of time, another employee received a document named GD20200909GAB31.doc with a job offer from the same company (creation date: 2020:09:14 07:50:00). By opening the documents from a potential employer, both victims activated malicious macros on their home computers. At the time of the attack, content was published on the domain that copied a page of the official website of General Dynamics Mission Systems, one of the world's largest manufacturers of military and aerospace equipment. The Lazarus Group had already used this brand in its attacks. The domain also had a valid SSL certificate."""

```

[4] doc = nlp(text)
entities = []
labels = []

*Figure 3: Importing & loading spacy NLP model*

The trained NER model then separates the entities it identifies from the raw text into labels from its dictionary.

```

doc = nlp(text)
entities = []
labels = []
position_start = []
position_end = []
for ent in doc.ents:
    entities.append(ent)
    labels.append(ent.label_)
    position_start.append(ent.start_char)
    position_end.append(ent.end_char)

df = pd.DataFrame({'Entities':entities,'Labels':labels, 'Position_Start':position_start,'Position_End':position_end })
df

```

	Entities	Labels	Position_Start	Position_End
0	(Termite)	ORG	166	173
1	(Kaspersky)	PERSON	213	222
2	(Termite)	ORG	248	255
3	(EarthWorm)	ORG	263	272
4	(Termite)	ORG	356	363
5	(BlackTech)	ORG	440	449
6	(Taiwan)	GPE	473	479
7	(EarthWorm)	ORG	498	507
8	(TenCent)	ORG	625	632
9	(a, University, Hospital)	ORG	647	668

	Entities	Labels	Position_Start	Position_End
0	(Termite)	ORG	166	173
1	(Kaspersky)	PERSON	213	222
2	(Termite)	ORG	248	255
3	(EarthWorm)	ORG	263	272
4	(Termite)	ORG	356	363
5	(BlackTech)	ORG	440	449
6	(Taiwan)	GPE	473	479
7	(EarthWorm)	ORG	498	507
8	(TenCent)	ORG	625	632
9	(a, University, Hospital)	ORG	647	668
10	(Guangzhou)	GPE	672	681
11	(China)	GPE	682	687
12	(Chongqing)	GPE	718	727
13	(China)	GPE	728	733
14	(the, end, of, September, 2020)	DATE	780	805
15	(2020:09:22, 03:08:00)	TIME	932	951
16	(2020:09:14)	CARDINAL	1103	1113
17	(General, Dynamics, Mission, Systems)	ORG	1353	1385
18	(The, Lazarus, Group)	ORG	1466	1483
19	(SSL)	ORG	1557	1560

Figure 4: Generating labels on the raw text

## Terminology:

ORG - organisation

GPE - Geopolitical entity

(Rest labels are self-explanatory)

The model is not perfect and has some errors e.g: Kaspersky is incorrectly labelled as PERSON whereas it should have been ORG.

Below is the visualisation of the entire text using **displacy**.

```
[5] displacy.render(doc, style='ent', jupyter=True)

Termite is a useful networking and penetration testing tool but we're seeing it used in attacks to enable access to machines too.

There has been little reporting on Termite ORG, beyond a brief mention in a report by Kaspersky PERSON of an earlier version of Termite ORG called EarthWorm ORG.

Below, we've provided an outline on some of the attackers we're seeing deploying Termite ORG. The same server has also previously been associated with a group known as BlackTech ORG that primarily targets Taiwan GPE.

We've also seen EarthWorm ORG packed in with malware from crypto-mining campaigns.

This malware is linked to a campaign previously reported on by TenCen ORG.

Machines at a University Hospital ORG in Guangzhou GPE China GPE, and a children's hospital in Chongqing GPE China GPE, were infected with crypto-mining malware.

At the end of September 2020 DATE, an employee of the pharmaceutical company received a document named GD20200909393903.doc with a job offer (creation date: 2020:09:22 03:08:00 TIME). After a short period of time, another employee received a document named GD20200909GAB31.doc with a job offer from the same company (creation date: 2020:09:14 CARDINAL 07:50:00). By opening the documents from a potential employer, both victims activated malicious macros on their home computers.

At the time of the attack, content was published on the domain that copied a page of the official website of General Dynamics Mission Systems ORG, one of the world's largest manufacturers of military and aerospace equipment. The Lazarus Group ORG had already used this brand in its attacks.

The domain also had a valid SSL ORG certificate.
```

*Figure 5: Visualising the results on raw text*

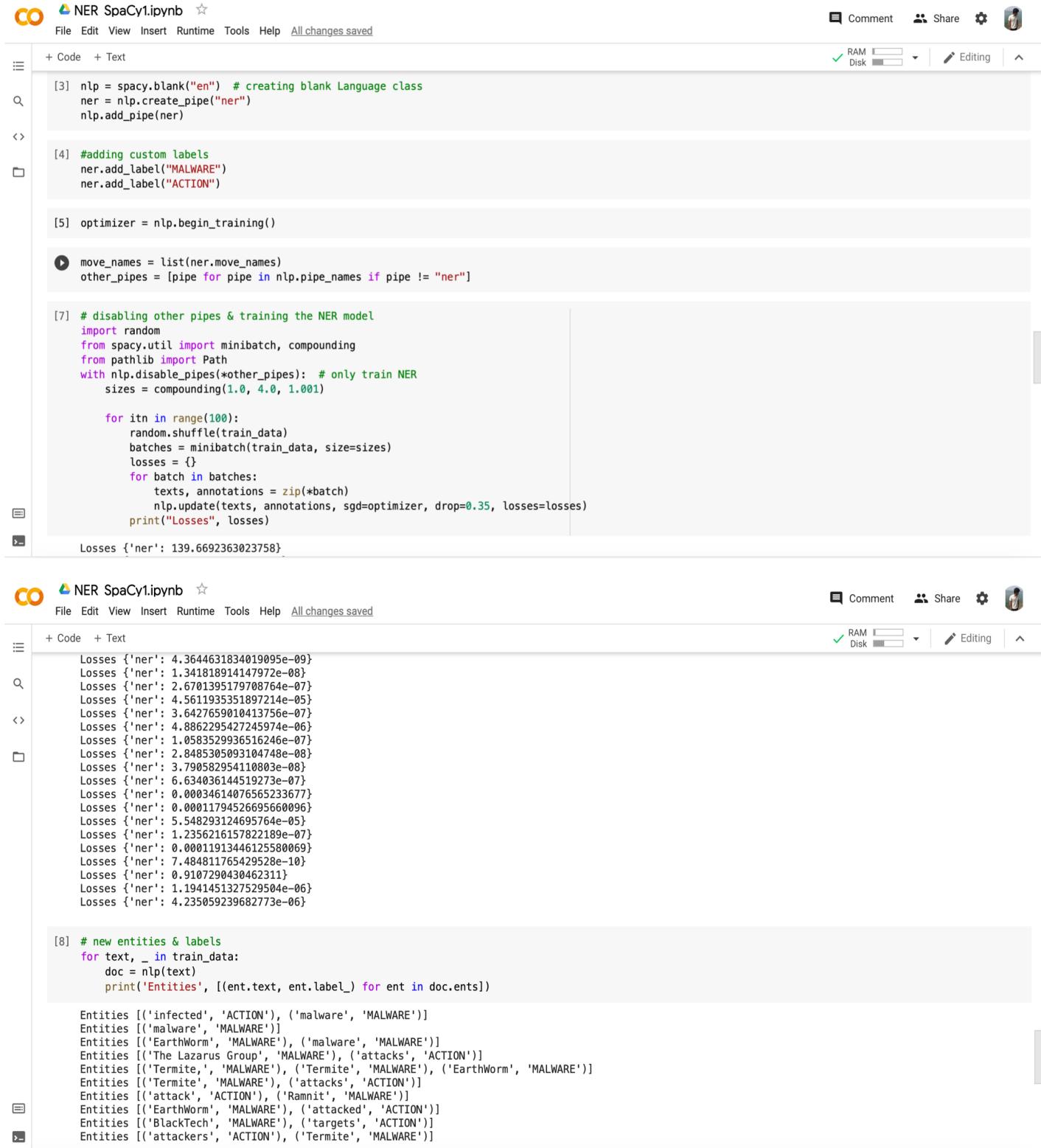
## ii) With custom labels for the datasets (with custom annotations):

I have defined a data dictionary with custom text annotations for various entities in the cybersecurity/digital forensics raw text that I extracted from the OTX Pulse.

I intend to train the NER model with the defined data dictionary, so that it can identify custom labels for entities that are otherwise not built into spaCy.

```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share
RAM Disk Editing
train_data = [
    ("Termite is a useful networking and penetration testing tool but we're seeing it used in attacks to enable access to machines too",
     {"entities": [(0,7,"MALWARE"),(88,95,'ACTION')]})�,
    ("There has been little reporting on Termite, beyond a brief mention in a report by Kaspersky of an earlier version of Termite called EarthWorm",
     {"entities": [(35,43,"MALWARE"),(117,124,"MALWARE"),(132,141,"MALWARE")]})�,
    ("Below, we've provided an outline on some of the attackers we're seeing deploying Termite",
     {"entities": [(81,88,"MALWARE"),(48,57,'ACTION')]})�,
    ("The same server has also previously been associated with a group known as BlackTech that primarily targets Taiwan",
     {"entities": [(74,83,"MALWARE"),(99,106,'ACTION')]})�,
    ("We've also seen EarthWorm packed in with malware from crypto-mining campaigns",
     {"entities": [(16,25,"MALWARE"),(41,48,'MALWARE')]})�,
    ("This malware is linked to a campaign previously reported on by TenCent",
     {"entities": [(5,12,"MALWARE")]})�,
    ("Machines at a University Hospital in Guangzhou China, and a children's hospital in Chongqing China, were infected with crypto-mining malware",
     {"entities": [(133,140,"MALWARE"),(105,113,"ACTION")]})�,
    ("By opening the documents from a potential employer, both victims activated EarthWorm macros and got attacked",
     {"entities": [(75,84,"MALWARE"),(100,108,'ACTION')]})�,
    ("At the time of the attack by Ramnit, content was published on the domain that copied a page of the official website of General Dynamics Mission Systems
     {"entities": [(29,35,"MALWARE"),(19,25,"ACTION")]})�,
    ("The Lazarus Group had already used this brand in its attacks",
     {"entities": [(0,17,"MALWARE"),(53,60,"ACTION")]})�
]
```

Figure 6: Defining the required dictionary



The screenshot shows two Jupyter Notebook sessions for a file named "NER SpaCy1.ipynb".

**Session 1 (Top):**

```
[3] nlp = spacy.blank("en") # creating blank Language class
ner = nlp.create_pipe("ner")
nlp.add_pipe(ner)

[4] #adding custom labels
ner.add_label("MALWARE")
ner.add_label("ACTION")

[5] optimizer = nlp.begin_training()

[6] move_names = list(ner.move_names)
other_pipes = [pipe for pipe in nlp.pipe_names if pipe != "ner"]

[7] # disabling other pipes & training the NER model
import random
from spacy.util import minibatch, compounding
from pathlib import Path
with nlp.disable_pipes(*other_pipes): # only train NER
    sizes = compounding(1.0, 4.0, 1.001)

    for itn in range(100):
        random.shuffle(train_data)
        batches = minibatch(train_data, size=sizes)
        losses = {}
        for batch in batches:
            texts, annotations = zip(*batch)
            nlp.update(texts, annotations, sgd=optimizer, drop=0.35, losses=losses)
        print("Losses", losses)

Losses {'ner': 139.6692363023758}
```

**Session 2 (Bottom):**

```
Losses {'ner': 4.3644631834019095e-09}
Losses {'ner': 1.341818914147972e-08}
Losses {'ner': 2.6701395179708764e-07}
Losses {'ner': 4.5611935351897214e-05}
Losses {'ner': 3.6427659010413756e-07}
Losses {'ner': 4.8862295427245974e-06}
Losses {'ner': 1.0583529936516246e-07}
Losses {'ner': 2.8485305093104748e-08}
Losses {'ner': 3.790582954110803e-08}
Losses {'ner': 6.634036144519273e-07}
Losses {'ner': 0.00034614076565233677}
Losses {'ner': 0.00011794526695660096}
Losses {'ner': 5.548293124695764e-05}
Losses {'ner': 1.2356216157822189e-07}
Losses {'ner': 0.00011913446125580069}
Losses {'ner': 7.484811765429528e-10}
Losses {'ner': 0.9107290430462311}
Losses {'ner': 1.1941451327529504e-06}
Losses {'ner': 4.235059239682773e-06}

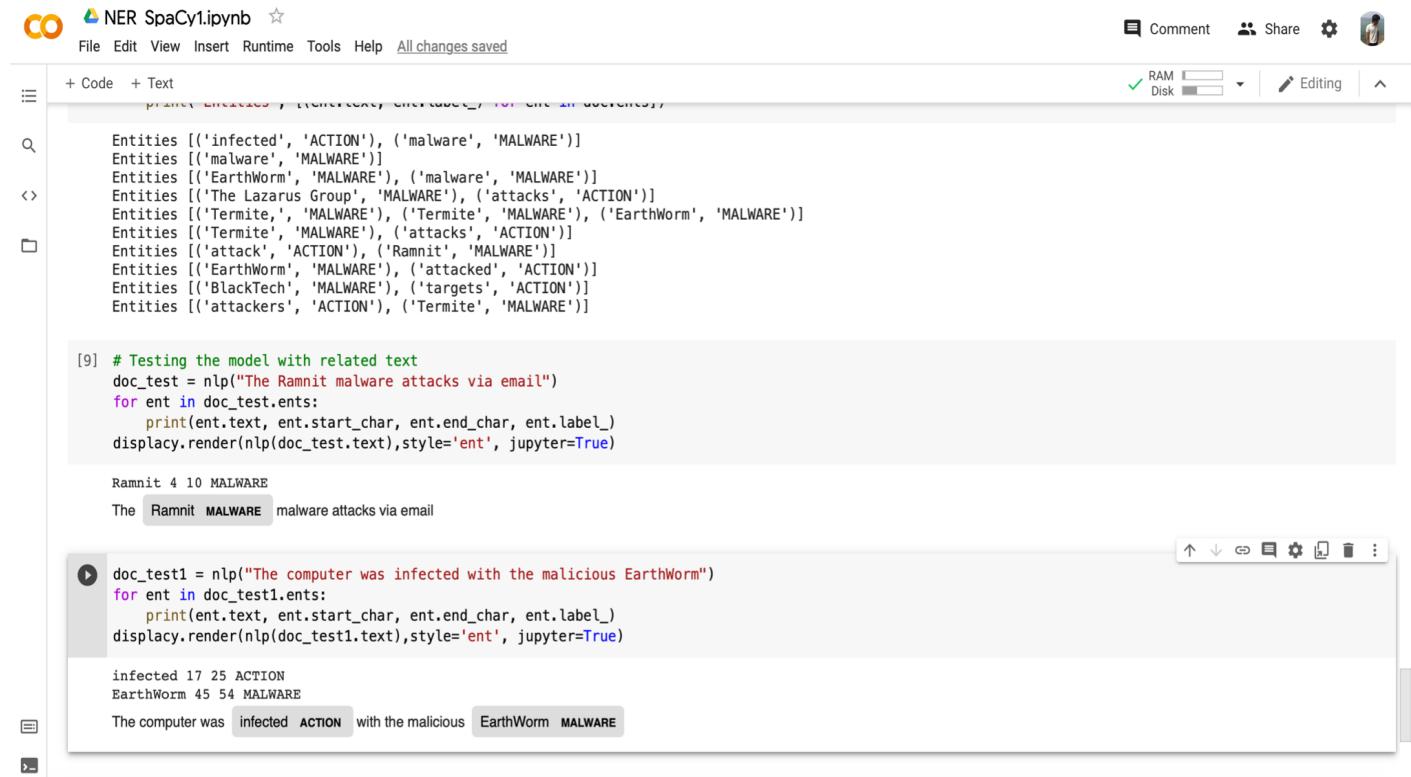
[8] # new entities & labels
for text, _ in train_data:
    doc = nlp(text)
    print('Entities', [(ent.text, ent.label_) for ent in doc.ents])

Entities [('infected', 'ACTION'), ('malware', 'MALWARE')]
Entities [('malware', 'MALWARE')]
Entities [('EarthWorm', 'MALWARE'), ('malware', 'MALWARE')]
Entities [('The Lazarus Group', 'MALWARE'), ('attacks', 'ACTION')]
Entities [('Termite', 'MALWARE'), ('Termite', 'MALWARE'), ('EarthWorm', 'MALWARE')]
Entities [('Termite', 'MALWARE'), ('attacks', 'ACTION')]
Entities [('attack', 'ACTION'), ('Rammnit', 'MALWARE')]
Entities [('EarthWorm', 'MALWARE'), ('attacked', 'ACTION')]
Entities [('BlackTech', 'MALWARE'), ('targets', 'ACTION')]
Entities [('attackers', 'ACTION'), ('Termite', 'MALWARE')]
```

Figure 7: Training the NER model with the dictionary

Below is the output that we get on the testing data,

The results are very close to what I expected:



The screenshot shows two Jupyter Notebook sessions. The top session, titled 'NER SpaCy1.ipynb', displays code and output for testing various malware entities. The bottom session, also titled 'NER SpaCy1.ipynb', displays code and output for testing different groups and their actions.

```

NER SpaCy1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM Disk Editing
[8] Entities [('infected', 'ACTION'), ('malware', 'MALWARE')]
Entities [('malware', 'MALWARE')]
Entities [('EarthWorm', 'MALWARE'), ('malware', 'MALWARE')]
Entities [('The Lazarus Group', 'MALWARE'), ('attacks', 'ACTION')]
Entities [('Termite', 'MALWARE'), ('Termite', 'MALWARE'), ('EarthWorm', 'MALWARE')]
Entities [('Termite', 'MALWARE'), ('attacks', 'ACTION')]
Entities [('attack', 'ACTION'), ('Ramnit', 'MALWARE')]
Entities [('EarthWorm', 'MALWARE'), ('attacked', 'ACTION')]
Entities [('BlackTech', 'MALWARE'), ('targets', 'ACTION')]
Entities [('attackers', 'ACTION'), ('Termite', 'MALWARE')]

[9] # Testing the model with related text
doc_test = nlp("The Ramnit malware attacks via email")
for ent in doc_test.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
displacy.render(nlp(doc_test.text), style='ent', jupyter=True)

Ramnit 4 10 MALWARE
The Ramnit MALWARE malware attacks via email

[10] doc_test1 = nlp("The computer was infected with the malicious EarthWorm")
for ent in doc_test1.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
displacy.render(nlp(doc_test1.text), style='ent', jupyter=True)

infected 17 25 ACTION
EarthWorm 45 54 MALWARE
The computer was infected ACTION with the malicious EarthWorm MALWARE

[11] doc_test2 = nlp("Termite is a DOS virus writer in high level language. It attacks Windows 95, 98 etc.")
for ent in doc_test2.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
displacy.render(nlp(doc_test2.text), style='ent', jupyter=True)

Termite 0 7 MALWARE
attacks 58 65 ACTION
Termite MALWARE is a DOS virus writer in high level language. It attacks ACTION Windows 95, 98 etc.

[12] doc_test3 = nlp("BlackTech is a cyber espionage group that targets East Asia, particularly Taiwan, and occasionally, Japan and Hong Kong")
for ent in doc_test3.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
displacy.render(nlp(doc_test3.text), style='ent', jupyter=True)

BlackTech 0 9 MALWARE
targets 42 49 ACTION
BlackTech MALWARE is a cyber espionage group that targets ACTION East Asia, particularly Taiwan, and occasionally, Japan and Hong Kong

[13] doc_test4 = nlp("The Lazarus Group has been active since at least 2009 and was responsible for the destructive attack against Sony Pictures Entertainment")
for ent in doc_test4.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
displacy.render(nlp(doc_test4.text), style='ent', jupyter=True)

The Lazarus Group 0 17 MALWARE
attack 94 100 ACTION
Pictures 114 122 MALWARE
The Lazarus Group MALWARE has been active since at least 2009 and was responsible for the destructive attack ACTION against Sony Pictures MALWARE Entertainment

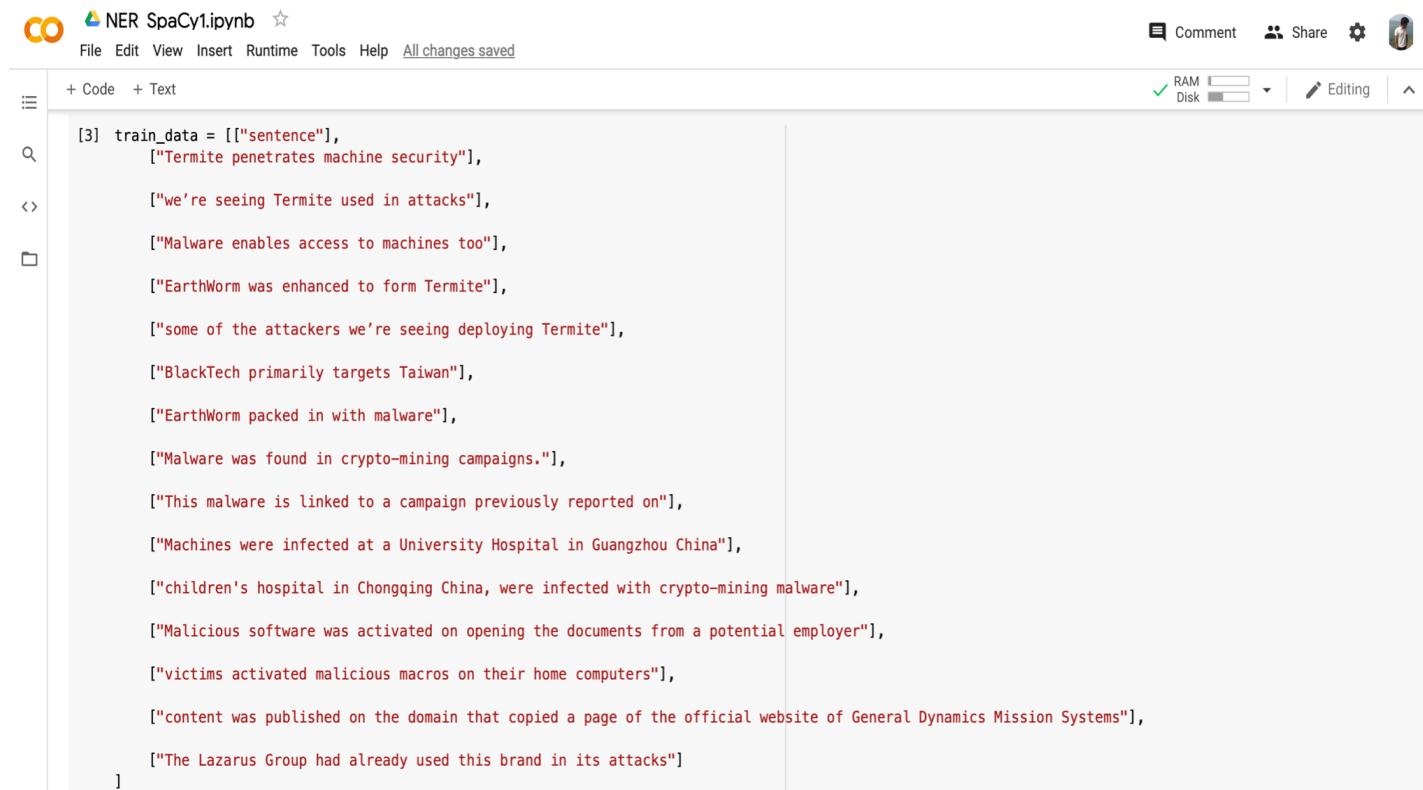
```

Figure 8: Testing the NER model with random texts

*The following is the updated work after uploading the above work on Vtop.*

**Step 3:** As we move towards building a **knowledge graph**, we refine the sentence segmentation of the data corpus collected. The sentences are segmented such that, there is one subject, one object & a ROOT to relate the two.

The subject of the sentence will act as source & object will act as the target with the root as the edge between source & target.



The screenshot shows a Jupyter Notebook interface with the title "NER SpaCy1.ipynb". The code cell contains the following Python code:

```
[3] train_data = [[{"sentence":  
                 ["Termite penetrates machine security"],  
                 ["we're seeing Termite used in attacks"],  
                 ["Malware enables access to machines too"],  
                 ["EarthWorm was enhanced to form Termite"],  
                 ["some of the attackers we're seeing deploying Termite"],  
                 ["BlackTech primarily targets Taiwan"],  
                 ["EarthWorm packed in with malware"],  
                 ["Malware was found in crypto-mining campaigns."],  
                 ["This malware is linked to a campaign previously reported on"],  
                 ["Machines were infected at a University Hospital in Guangzhou China"],  
                 ["children's hospital in Chongqing China, were infected with crypto-mining malware"],  
                 ["Malicious software was activated on opening the documents from a potential employer"],  
                 ["victims activated malicious macros on their home computers"],  
                 ["content was published on the domain that copied a page of the official website of General Dynamics Mission Systems"],  
                 ["The Lazarus Group had already used this brand in its attacks"]}]
```

The subject and object entities of the sentences are extracted.

The screenshot shows a Jupyter Notebook interface with a single code cell. The cell contains Python code that processes a sentence to extract subject and object entities. The code uses SpaCy's dependency parser to identify words based on their grammatical role ('subj' or 'obj') and concatenates them into larger entities if they are part of a compound word. The code then returns a list of these entities. A command cell below runs the function on the sentence "Malware enables access to machines too" and outputs the result: ['Malware', 'machines'].

```
if tok.dep_ == "compound":  
    prefix = tok.text  
    # if the previous word was also a 'compound' then add the current word to it  
    if prv_tok_dep == "compound":  
        prefix = prv_tok_text + " " + tok.text  
  
    if tok.dep_.endswith("mod") == True:  
        modifier = tok.text  
        # if the previous word was also a 'compound' then add the current word to it  
        if prv_tok_dep == "compound":  
            modifier = prv_tok_text + " " + tok.text  
  
        if tok.dep_.find("subj") == True:  
            ent1 = modifier + " " + prefix + " " + tok.text  
            prefix = ""  
            modifier = ""  
            prv_tok_dep = ""  
            prv_tok_text = ""  
  
        if tok.dep_.find("obj") == True:  
            ent2 = modifier + " " + prefix + " " + tok.text  
  
            prv_tok_dep = tok.dep_  
            prv_tok_text = tok.text  
  
    return [ent1.strip(), ent2.strip()]
```

```
get_entities("Malware enables access to machines too")  
['Malware', 'machines']
```

✓ 0s completed at 9:51 PM

Next the relations are extracted to form the edges of the Graph using spaCy's Matcher, which lets you find words and phrases using rules describing their token attributes.

The pattern defined in the function tries to find the ROOT word or the main verb in the sentence.

The screenshot shows a Jupyter Notebook interface with the following details:

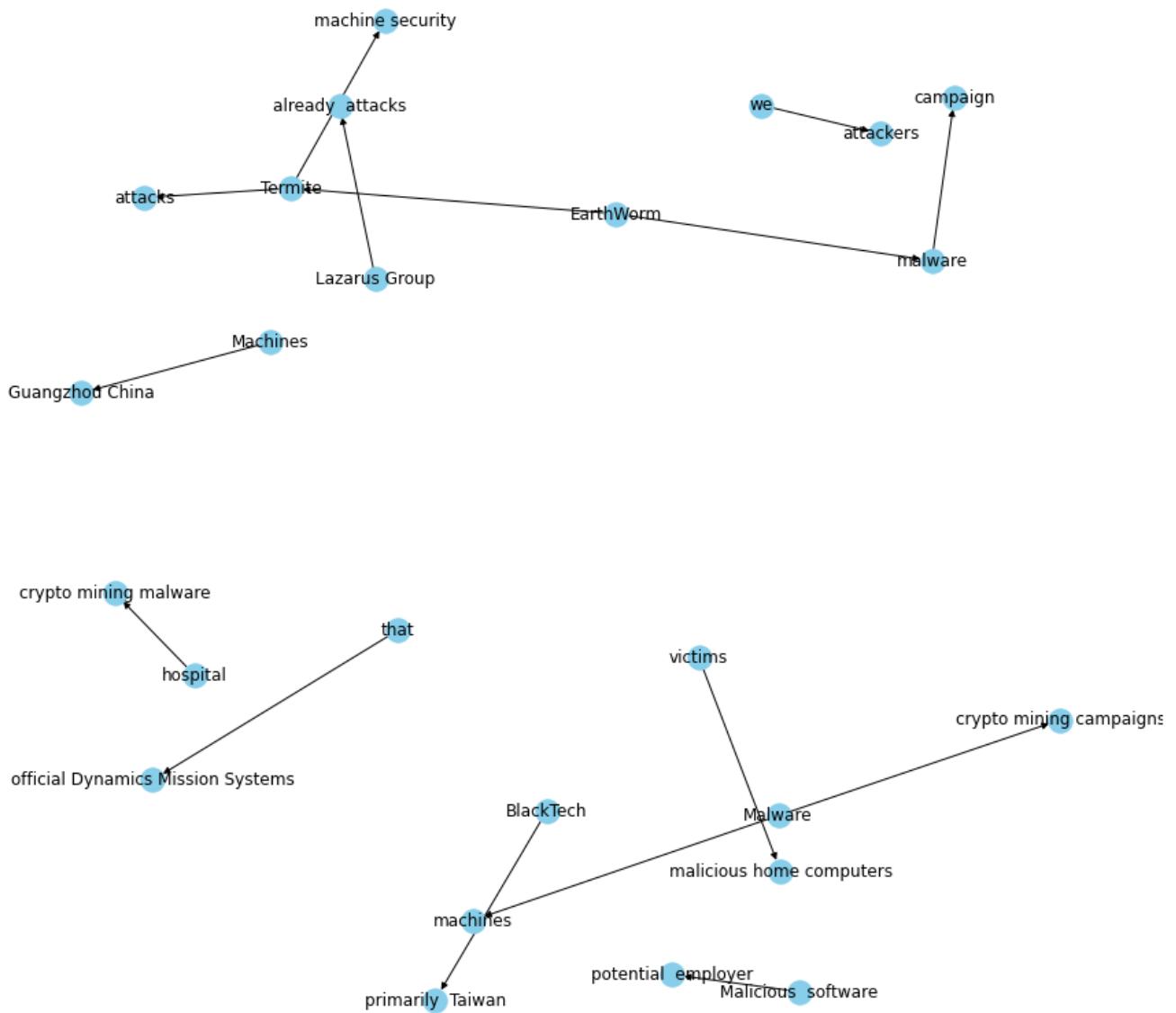
- Title:** NER SpaCy1.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved
- Code Cell:** Contains Python code for defining a Matcher and extracting relations from a sentence.

```
def get_relation(sent):
    doc = nlp(sent)
    matcher = Matcher(nlp.vocab)
    pattern = [{"DEP": "ROOT"}, {"DEP": "prep", "OP": "?"}, {"DEP": "agent", "OP": "?"}, {"POS": "ADJ", "OP": "?"}]
    matcher.add("matching_1", None, pattern)
    matches = matcher(doc)
    k = len(matches) - 1
    span = doc[matches[k][1]:matches[k][2]]
    return(span.text)
```

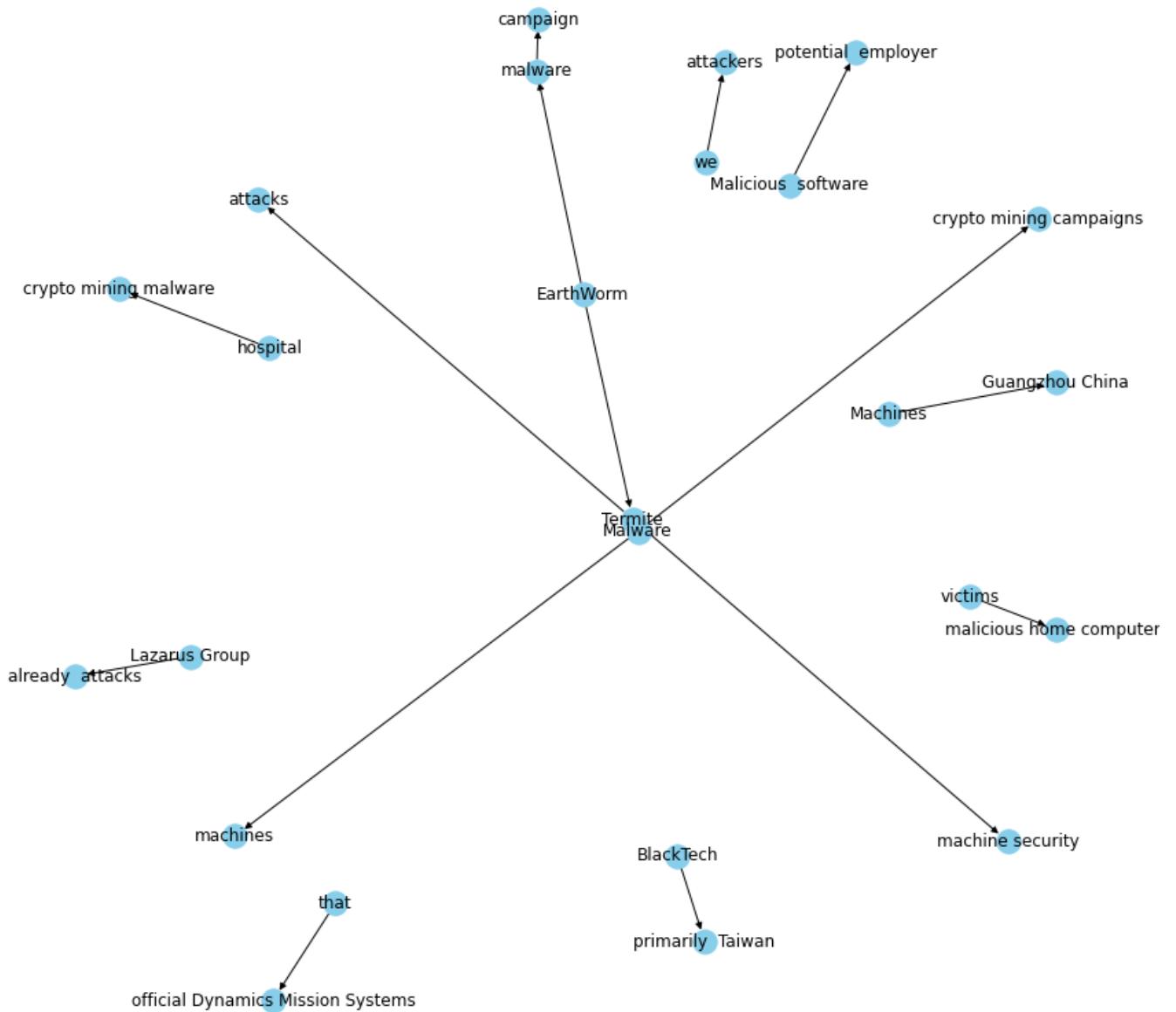
- Output Cells:**
  - [11] get\_relation("BlackTech primarily targets Taiwan")  
'targets'
  - [12] relations = [get\_relation(i) for i in tqdm(df["sentence"])]  
100%|██████████| 15/15 [00:00<00:00, 21.32it/s]

Finally, using the *networkx* library, created a network from the dataframe in which the source, target nodes & the relation edges are structured & hence the knowledge graph. The **tuned graph** is on the next page.

The nodes represent the entities, and the edges or connections between the nodes represent the relations between the nodes.



Following is the result of tuning the knowledge graph formation:



I have done the tuning in a separate colab file, so as to not alter the previous work:

<https://colab.research.google.com/drive/1W6zPpdAHuCgK9FsTDhsYFdBqKnQoG2?usp=sharing>