Article: ***Named Entity Recognition and Relation Extraction: State-of-the-Art***
https://dl.acm.org/doi/abs/10.1145/3445965?casa_token=oLYI6UI4vjkAAAAA:b_yS0qvgtDbXTwYGfGgWREeBblvVTbN4zBsbZ7Sx_J0ouVg6D7geST7J-XPc5Zcekn3w0XWi_plY

Q1: *Can you summarize the state of the art you learned in no more than one page?*

The article aims to propose an alternative state-of-the-art method of relational extraction and NER machine learning.

The article first begins by explaining the current methodologies of NER/RE and the background of NER/RE. The authors gathered data from past surveys and conferences of current NER models and important information about the current state of NER such as the datasets used to train the models, the general tasks completed, and the language the model uses. The article then compares the differences in the growth of NER as compared to RE which shows that NER has become more prevalent in recent years with many more surveys being conducted on NER than RE.

Throughout the rest of the article, the authors analyze various ways/methods of training NER models for English, non-English, and specified domains such as neural networks, un/supervised approaches, CRF, and rule-based approaches. The article explores how each approach accomplishes its expected task and the effects they each can have on performance, referencing some of the earlier surveys and implementations of the approach as mentioned previously.

In its conclusion, the article states that one particular approach that has been seen to improve performance is employing joint models. This includes combining 2 or more of the previously explored approaches and using them to train the model accordingly. It is explained that using a joint model approach can reduce the errors made by modeling multiple problems simultaneously. These joint models can make use of multiple machine-learning processes at the same time making them more accurate and precise.

Q2: *Can you present your implementation and how it works? Are you able to reproduce the results of the paper? Observe, analyze and interpret the results.*

I tried to implement the joint model approach as described in *Named Entity Recognition and Relation Extraction: State-of-the-Art.*

My implementation uses the SpaCy library's base English Model. To change the functionality, I used the entity ruler and pipelining to add a rule to the model so that it can recognize certain diseases. I also attempted to follow this tutorial to train the model, but in

the interest of time, I chose to stick with the entity ruler only. The base English model by SpaCy uses a CRF approach to train its model as described by this article. To modify this model, I then used a rules-based approach to train the model to recognize certain words as "DISEASE" words. I implemented this using SpaCy's entity ruler. This way, I was able to use my model to extract all entities related to disease names.

I stored my model on a Django backend server, which is used by my Chrome extension. I tested this implementation using various methods, one of which was by printing out the results of the model. I did this after implementing the model for the first time by running the model over a text file that contained information about COVID-19. The model worked as expected, being able to extract all the instances of COVID-19, Coronavirus, etc. from the text. One unexpected result from this, however, was that some diseases, like Diabetes, aren't caught by my model. This is because the training dataset for my model is very small so some diseases weren't labeled properly and thus weren't caught.

Q3: *Can you present your extension and how it works? Can you make it more useful?*

The extension works by getting all the text and automatically sending it to the backend server using a POST request once the webpage loads. After receiving the request, the model identifies all the diseases that are present in the text and adds them to a list. Once the user presses the highlight button on the extension, this sends a GET request to the server which sends the list of identified diseases back to the extension. The extension then highlights the diseases on the webpage.

Due to the time constraints of this assignment, I chose to only highlight the diseases, but there are many ways to make the extension more useful. One way is to identify the symptoms associated with each disease by adding a symptoms-based rule and then highlighting the symptoms in a different color. This way, users can see which symptoms are associated with each disease.

Part 4:

- **Q4.1:** *What have you learned from the exercise above?*

I learned a lot about NER/RE and how to implement state-of-the-art techniques to make a useful application. In particular, I learned a lot about the various techniques of training NER models which were previously unfamiliar to me, and the benefits of each technique. I was familiar with some of the methods, such as neural networks, but it was exciting to learn more about joint modeling, CRF, rules-based modeling, and more. I also learned how to

connect and use the model from an application perspective and how to send/receive data from a Django server that hosted my model after training it.

- **Q4.2:** *Do you think the entity extraction results were satisfactory for practical usage? What can be improved?*

While I am proud of the progress and results I achieved, I believe that much more can be done to make the results better for practical usage. While the extraction of certain disease names is decent, the overall dataset that I used in my rules-based approach is far too small. As a result, my model can only recognize a handful of disease names. Additionally, I wasn't able to add recognition of other types of entities, such as symptoms, which was one of my goals. Adding this functionality and using larger and more robust datasets would be very beneficial for the practical usage of entity extraction in my application.

- **Q4.3:** *What new applications can be built using NER? Propose a cool application you can build that does not exist today.*

One application idea I have for NER is how it can be used in computational biology and medicine. I know that it is very common in computational biology and the medical field to classify proteins, which are often represented in the English language as 3 letter abbreviations such as Gly (Glycine) or Arg (Arginine). Technology exists for the analysis of medical text to create graph structures, but I think it would be cool to use NER to analyze protein chains. This can be applicable in quickly finding abnormalities or other insights to predict things such as chronic conditions, new mutations, etc.

Citations/References:

- https://dl.acm.org/doi/pdf/10.1145/3445965
- https://ner.pythonhumanities.com/intro.html
- https://developer.chrome.com/docs/extensions/mv3/content_scripts/
- https://developer.mozilla.org/en-US/docs/Web/API/NodeList
- https://developer.mozilla.org/en-US/docs/Web/API/Document_object_model/Locating_DOM_elements_using_selectors
- https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelectorAll
- https://developer.chrome.com/docs/extensions/mv3/getstarted/tut-focus-mode/#step-4
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
- OpenAI. (2023). *ChatGPT* (Nov 19 version) [Large language model]. https://chat.openai.com/chat