Project architecture and detailed workflow

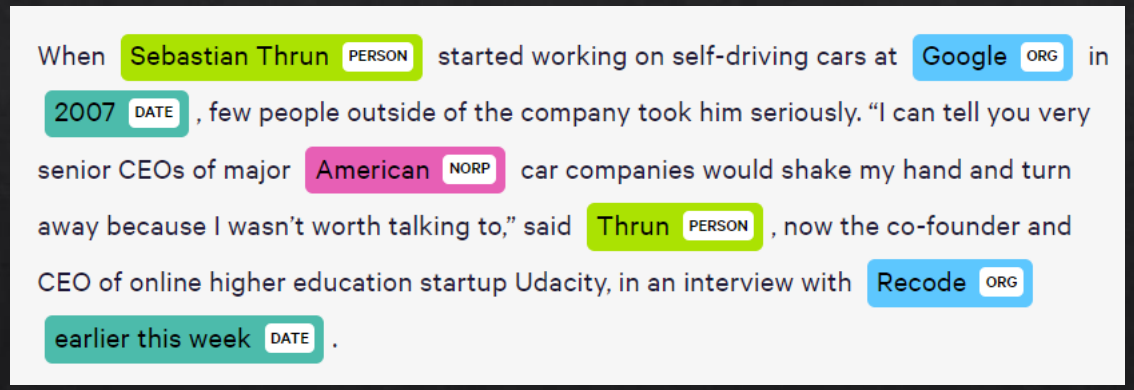
Astha

NLP

- A sub-field of AI, enabling computers to understand and process natural language (in any form - speech or signing, used by humans).
- NLP helps extract useful data from raw, unstructured human language.
- For e.g., for identifying interesting patterns in clinical texts and electronic health records, predicting the development of certain diseases based on the data mined from these texts.
- open source Python libraries like spaCy, textacy, and neuralcoref are used for applying NLP.

NER

- A subfield of NLP is Named Entity Recognition (NER) in which individual terms are identified within text and tagged with specific labels.
- A typical NER system can tag people's names, company names, places, product names, dates, times, money amounts, event names, etc.



- These models not only do a dictionary look-up, but instead they use the context of how a word appears in the sentence and use a statistical model to guess which type of noun a word represents.
- So a good NER system will differentiate between the person Tesla and the company Tesla.

GOAL

- **Apply NER technique to scientific papers to help researchers find articles of interest.**

- ***Why?***

  - NER labels can be useful for searching quickly through multiple texts since one doesn't need to use the exact term to find a relevant result.
  - Additionally, by using NER to extract named entities, it can be easier to categorize and organize articles based on the entities they mention. This can facilitate the discovery of new relationships and patterns in the data, which can help to identify new areas of research and further insight.

How NER works?

- When an article is indexed in PubMed, its title, abstract, and full text are analyzed by an NER system.
- The system identifies named entities in the text, such as disease names, gene names, and so on. These named entities are then associated with the article and stored in a database.
- When a user conducts a search on PubMed, they can use these named entities as search terms, rather than having to use the exact terms found in the article's title or abstract.
- For example, if a user is interested in finding articles about breast cancer, they can use the named entity "breast cancer" as a search term, rather than having to search for every possible variation of the term ("breast carcinoma," "invasive ductal carcinoma," etc.).

- This way, the search results will include articles that mention "breast cancer" even if they don't use that exact term in the title or abstract.
- The NER system doesn't necessarily "know" that "cancer" and "carcinoma" are the same thing, but it can be trained to recognize them as variations of the same named entity.
- This is done through the process of entity linking or semantic normalization, where different variations of a named entity, such as synonyms, acronyms or abbreviations, are mapped to a unique identifier.
- The accuracy of this linking is highly dependent on the quality of the training data and the algorithm used for this task.

WORKFLOW

1. Create a corpus of scientific abstracts using the NCBI E-utilities API

   - Download abstracts from PubMed using the PubMed API or Entrez Programming Utilities. Along with Bio.Entrez module in Python (esearch and efetch functions).

     - esearch(db="pubmed", term="biological[All Fields]", mindate and maxdate parameters or datetime module)
     - retmax parameter specifies the maximum number of records to retrieve. Check for the tool you are using.
     - efetch() – mention xml or text format (text is less structured than xml)
     - xml_data will contain a single string with all the XML records concatenated together. Each individual record is separated by a newline character.
     - If you want to work with the records individually, you'll need to parse the string and extract the individual records. You can use a library such as xml.etree.ElementTree in python to parse the data and access the elements of the XML individually.
     - Parse and store abstract portion (also keep a dict of PMID, Date of publication)

- Download abstracts from PubMed using the Europe PMC RESTful Web Service or PubTator Central API
- By requests lib
  - URL, specify endpoints in the url (query = “biological”, mindate, maxdate), requests.get(), data = response.json()

Note:

after downloading the abstracts from PubMed, you can filter them according to a specific term or field using string manipulation and/or natural language processing techniques in Python

2. Apply NER technology to tag words within the abstracts with specific keywords
    - Use libraries like spaCy, NLTK (POS Tagging), or StanfordNLP
    - Focus on pre-trained named entity recognition (NER) models for biological documents in Python (BioBERT, SciBERT, BioNLP-ST, NEEL, NCBI-BERT via Hugging Face's transformers library and NLTK lib).
    - Load pre-trained NER model from the lib
    - Pre-processing and data cleaning (removing special characters, punctuations, stop words, numbers, email addresses, urls, HTML or XML tags, duplicates; lowercasing text, tokenization)
    - Split abstracts into train, test, validation sets

- Fine tune by getting NER pipeline components and adding new (specific to biological) entity labels to the NER pipeline (according to our train data)
- Training model on train set
- NER model performance (precision, recall, F1, confusion matrix) on test set
- As per scores, choose model
- **Apply trained model to extract entities and their labels from unseen abstracts**
  - Iterate through abstracts and process each one with the NER model
  - Iterate through the entities in each abstract
- **Can return entities, their labels as a dict. Also add metadata from xml – abstract text, abstract ID, date, etc.**
- Way to store this info as cached files or DB
- Create DB - As NER info will be queried and searched, DB is more efficient option and serves better organization of data
- DB – Abstract text, PMID, date of publication, entities, their labels

3. Generate functions to query your tagged corpus stored in a database using a keyword or phrase and return informative results

- Keyword – keyword/phrase you want to search for in the corpus (input specific gene symbols (e.g. IL2, IFNA) or phrases (e.g. "cell cycle" or "apoptosis")).
- Use sqlite3 or sqlAlchemy
- Define function with parameters – keyword and db_path
- Execute the query to retrieve the relevant rows from the table where the abstract text column contains the keyword
- Return corresponding labels, PMID, date from the rows as a dict **that can be used in task 4****
- Modify to include a date range restriction
  - Add 2 optional parameters - start_date and end_date
  - restrict the searches to a range of dates by adding a WHERE clause to the SQL query that filters the results based on a date range.
  - Retrieve all rows from the table where the abstracr text column contains the keyword and the date is within the given range

Packaging

- Package contains
  - Raw code
  - CLI for all functions
  - Unit tests
  - Readme
  - Setup python file with package requirements

4. Create a frontend which allows one to search the corpus using a keyword or phrase.
    - Jinja or CSS and JavaScript templates
    - Run.py – functions to get and post info
    - Input box to enter keyword/phrase
    - implement a date range filter in a front-end application using HTML, CSS, and JavaScript:
        - by creating a form that contains two input fields for the start and end date, and a button to submit the form.

    - Table (call functions from task 3* in run.py to fetch this info from DB)
        - PMID
        - Hyperlink to pubmed page – need to add to DB (maybe from xml in task 1 or alternative ways)
        - Date of publication
        - Entire abstract
        - ***A list of keywords/phrases in the abstract which were identified as being tagged with the user's search query***
        - Download this info in available formats (check formats)

Containerization

- Docker file
- Probable division:
  - Tasks
    1. Creating corpus -
    2. NER model -
    3. + DB, Function to query DB -
    4. front-end** -

    My preference – NER Model or Front-end development ☺

    1. CLI (5) -
    2. Unit tests (5) -
    3. Packaging (5) -
    4. Containerization (docker file) -

    Confirm division with Bruce