Annotating Cancer Pathology Reports to Facilitate the Training of a Natural Language Processing Model

by

Rhys Smith

Submitted in partial fulfillment

of the requirements for the degree

of

Bachelor of Science (Honors) in

Computer Science

Irving K. Barber Faculty of Science

The University of British Columbia - Okanagan

Kelowna, British Columbia, Canada

Abstract

Natural language processing (NLP) is a form of artificial intelligence that allows computer models to rapidly analyse large volumes of complex text and summarize essential information. In healthcare, NLP can assist medical professionals with extracting key information from medical records to support clinical decision making. To achieve this, an NLP model must be trained to identify the critical medical information. This project explores the potential for training an NLP model to process breast cancer pathology reports.

The starting point in this process was to manually annotate breast cancer pathology reports from BC Cancer data to standardize and categorize pathology information. The pathology reports were initially sorted by cancer type to ensure that only breast cancer pathology reports were selected for annotation. The three annotation software programs used were Prodigy, Doccano, and Label-Studio to help create the annotated breast cancer dataset. This dataset included categories for patient information, cancer stage, biopsy characteristics and other relevant details. The outputs of each annotation software have been converted to a standardized data type to allow for easier training of an NLP model.

Annotating and converting breast cancer pathology data for this project is the first step in reaching the goal of developing a medical NLP model capable of summarizing critical pathological information for medical professionals.

Table of Contents

Introduction

1.1 Background

In recent years, advancements in technology have positioned Artificial Intelligence (AI) as a valuable tool in healthcare, with applications ranging from assisting with medical imaging analysis to automating administrative work. Among these tools, Natural Language Processing (NLP), a subfield of AI, has become an increasingly important asset in healthcare. By using machine learning to interpret human language, NLP allows computers to analyze unstructured text by classifying, extracting, or summarizing key information (Stryker and Holdsworth, 2024). Cancer pathology reports contain critical information for clinical decision-making. The reports often provide detailed insights into tumor characteristics, biopsy results, and other essential diagnostic elements. Given the complexity of these reports, NLP can be valuable in efficiently extracting and categorizing pathological information. However, these documents are often lengthy with unstructured content, making it time-consuming for medical professionals to review the reports. A well-trained NLP model has the potential to reliably extract crucial information from cancer pathology reports and summarize key details to allow medical professionals to analyze and understand patient information more efficiently.

To develop an NLP model capable of effectively processing pathology reports, it is necessary to create an annotated dataset that can be used to train the model. A poorly trained NLP model would struggle to extract pathology data, as it would lack the necessary understanding of medical terminology and context. This increases the likelihood of errors or incomplete information extraction. Manually annotating pathology reports is the first and most important step in the process of training an NLP model. Annotating establishes standardized categories and labels to teach the NLP model to accurately identify critical information and learn from it. For this project, we focused on annotating breast cancer pathology reports from the BC Cancer recurring cancer dataset, consisting of 18,488 pathology reports from 7,519 patients. For the annotation process, we used three software programs: Prodigy, Doccano, and Label-Studio. Each tool was used to annotate between 10 and 22 breast cancer pathology reports, contributing to the annotated dataset by categorizing relevant medical details.

1.2 Problem Statement

Despite the potential NLP has in supporting medical professionals, currently available pre-trained NLP models such as medspaCy and CAN-BERT (Clinical Assertion and Negation Classification Bidirectional Encoder Representations from Transformers) have shown to be lacking in their ability to accurately identify and summarize medical data (Su, 2024). Previous work has found that existing medical NLP models also perform less than optimal when attempting to extract information from cancer pathology reports (Santos, 2022).

To address this challenge, a dataset of annotated breast cancer pathology reports was created by manually annotating 43 reports from the BC Cancer recurring cancer dataset, with help from a second-year medical student. This ensured the details within the reports was accurately identified to make annotated data suitable for NLP model training. Three different annotation tools were used to evaluate the accuracy of the outputs and ease of use of each tool to determine the best option for future annotation work for this project. This annotated dataset will serve as the foundation for continued annotation work, followed by training the NLP model to evaluate its precision and accuracy.

1.3 Objectives

The primary objective of this project was to explore and evaluate various annotation tools to create a structured and annotated breast cancer pathology dataset. This annotated dataset is the initial step required in training an NLP model.

The specific objectives for this project include:

1. **Sorting Pathology Reports by Cancer Type** – Standardize and categorize pathology reports based on cancer type and patient health number (PHN).
2. **Comparison of Annotation Tools** – Setup and compare the usability of available annotation software programs. Evaluate and decide on three annotation tools to use for the project.
3. **Manual Annotation of Reports** – Manually annotate a portion of breast cancer pathology reports from the BC Cancer dataset using Prodigy, Doccano, and Label-Studio as annotation tools.
4. **Data Conversion between Annotation Tools** – Convert annotation outputs into a standardized format, compatible with all three tools, to help support NLP model training.

1.4 Significance of the Study

Using NLP to extract key clinical information from cancer pathology reports can save medical professionals time by summarizing relevant details. Medical professionals currently spend significant time reviewing information from unstructured pathology reports. Automating this process through NLP has potential to reduce the time medical professionals spend reviewing reports. This could allow medical professionals to spend more time with patients while also helping identify information that may inadvertently be overlooked.

This study is significant because it establishes the annotation guidelines and foundational dataset needed to develop an NLP model for extracting important details from cancer pathology reports. By converting the annotated pathology data to a standardized format, we can efficiently train the NLP model, regardless of the annotation software used. This research contributes to the field of medical informatics, the application of information technology and data science to improve healthcare, by establishing an annotation guideline for future studies and evaluating the available annotation tools that best support this project.

1.5 Scope and Limitations

This study specifically focuses on annotating breast cancer pathology reports from the BC Cancer dataset. Through the annotation process, the second-year medical student and I categorized important patient information such as: cancer staging, biopsy characteristics, and other relevant pathology details. Three annotation software programs, Prodigy, Doccano, and Label-Studio, were used for this task to compare their usability, the quality of the output data, and to identify the tool best suited for continued annotation work.

While this study lays the groundwork for training an NLP model, the actual development of the model is beyond the scope of this project. Additionally, the annotation process relies on human expertise, which introduces potential limitations, such as variability in annotators' decision-making and time constraints, potentially limiting the number of reports that we were able to annotate. Efforts have been made to ensure consistency across annotated reports through our annotation guidelines.

1.6 Conclusion

This project aimed to provide the initial steps for training an NLP model by developing annotation guidelines for future researchers and contributing to the creation of an annotated breast cancer pathology report dataset. Through using multiple annotation tools and allowing the conversion between annotated pathology data, this research contributes to the broader field of medical informatics. The findings from this study will serve as an important step toward developing an NLP model capable of summarizing critical pathological information, ultimately supporting medical professionals with patient care.

Review of Literature

2.1 Introduction

Through technological advancements, Natural Language Processing (NLP) has become an important part of people's everyday lives. With the widespread availability of voice assistants and text prediction tools to help with asking questions, getting directions and much more, people rely on NLP technologies in their everyday lives often without fully understanding what NLP is or how it works (Neupane, 2023). Natural Language Processing, or NLP, is a subfield of AI (Artificial Intelligence) that uses machine learning to process and analyze text through a series of tasks called a pipeline (Macri et al., 2023). The NLP process begins with pre-processing large amounts of text data by filtering any misspelled or unimportant words such as “the” or “is.” This is followed by tokenization where each sentence is broken down into smaller, more manageable parts for analysis. After tokenization, the NLP model then analyzes and learns from these tokens, gradually developing the ability to differentiate and classify text within the context of the request. This is similar to how we interpret meaning within the text we read and write (Nadkarni et al., 2011).

The ability for an NLP model to analyze and extract text has led to many possibilities within the medical field. With advanced AI technology now available, we can create and train NLP models to help make doctors and other medical professionals' lives easier (Hao et al., 2021). One of the difficulties medical professionals face is the ability to extract information and understand patient history from past pathology reports in a timely manner (Reiner, 2015). This can be for a number of reasons such as: each doctor’s unique writing style, differing institutional standards from clinics to hospitals and rushed pathology reports leading to scattered and unfocused notes making it difficult to understand for other doctors on the care team.

NLP software has the ability to reduce the time needed for doctors to analyze patient pathology reports and quickly catch up on a patient's current care plan, family history, and other essential details. The process begins by manually annotating a large number of pathology reports to classify essential patient details. Once fully annotated, the dataset can then be used to train an NLP model to recognize and distinguish patient details within new, unannotated reports (García Barragán, 2023).

Identifying the most suitable annotation tools is an important step in the NLP process. This review provides an in-depth examination of three NLP annotation tools, along with an overview of other available tools that may be suitable for this project. BRAT (BRAT Rapid Annotation Tool), Prodigy and SparkNLP all of which have been previously used in academia for similar tasks within the medical field. The three different annotation tools represent different approaches to annotation: BRAT is used through a web-based user interface with an emphasis on relationships and associations between annotated items (Stenetorp et al., 2012). Prodigy emphasizes an easy-to-use web-based user interface with the ability to easily create entities to be used for annotations. Prodigy comes included with spaCy, an integrated NLP tool that simplifies the NLP training process after annotation. (Montani & Honnibal, 2017). SparkNLP is an all-in-one tool that uses pre-existing datasets to streamline data annotation, while also offering manual annotation capabilities through its Generative AI Lab. SparkNLP aims to be one of the most efficient and time-saving annotation solutions available. However, Generative AI Lab does require a subscription to use (“Generative AI Lab,” n.d.). Each annotation tool has its own strengths and limitations, allowing for exploration of each one to determine which tools may best suit the needs of this project.

This review will examine the characteristics, functionalities, and applications of BRAT, Prodigy, SparkNLP, and other potential tools, providing a comparative analysis to highlight each tool's strengths and limitations. By analyzing these tools, this review aims to improve the understanding of the available annotation tools to assist with medical records analysis and annotation.

2.2 Review of Literature

BRAT (BRAT Rapid Annotation Tool): BRAT (BRAT Rapid Annotation Tool) was introduced in 2011 as one of the first widely accessible and free to use annotation tools, featuring a user-friendly interface that made it easy for users of all experience levels to start annotating (Stenetorp et al., 2012). Over time, it has become an important tool in natural language processing (NLP) workflows due to its accessibility and ease of use (Patil, 2022). BRAT’s primary advantage is its accessibility, enabling users with minimal technical experience to annotate text easily through a mouse-based interface similar to conventional text editors (*Features - Brat Rapid Annotation Tool*, n.d.). BRAT is an open-source project allowing contributions from Python developers, although it has not seen active development in recent years. The latest official release of BRAT was in 2012, and its last update on GitHub was in 2021, indicating limited ongoing support (Pyysalo et al., 2011/2021).

Despite the limited support, BRAT is still being used in healthcare settings as recently as early 2024 to annotate clinical concepts from electronic health records (EHR) for easier extraction (Fu et al., 2024). BRAT is also highly customizable software, allowing users to easily label entities, relationships, and events. This capability is particularly useful for identifying specific entities within pathology reports, such as “symptom,” “treatment,” and “diagnosis.” It is also helpful for defining relationships to link past symptoms to current treatment plans, assisting doctors in analyzing potential cause-and-effect patterns (Stenetorp et al., 2012). BRAT’s support for various text file formats and export options ensures it is compatible with different types of pathology reports.

BRAT does have several limitations affecting its ability to handle large datasets of medical information. As a fully manual annotation tool, where each annotation must be made by hand, BRAT can be challenging to use with large-scale datasets, such as those in the medical field that often contain thousands of records. Manually annotating each record can be a time-consuming and labor-intensive process as BRAT lacks the built-in NLP learning features that could streamline annotation. Without such automation, projects using BRAT may require significantly more time and resources, as every annotation must be entered. Additionally, BRAT’s installation process presents usability barriers. BRAT requires a Linux server and cannot be installed on Windows systems without an external application. This can be challenging for new users unfamiliar with Linux or command-line interfaces. BRAT’s interface appears outdated because it has not received a major update in over a decade (Pyysalo et al., 2011/2021). The outdated system makes it difficult for users to modify entities, which are managed through a separate system folder rather than within the application itself. This adds complexity to the tool, especially when compared to Prodigy, where entities are defined during setup, simplifying configuration (Montani & Honnibal, 2017).

While BRAT lacks the automation and scalability features of newer tools, it remains a valuable annotation software due to its adaptability and open source, free-to-use nature. Its customization options and interactive user interface (UI) make it a solid choice for projects requiring entity and relationship annotations on a limited budget. Despite its outdated UI and limited scalability, BRAT continues to be used in NLP research and applications where reliability and customization are key.

Prodigy: Prodigy, developed by Explosion AI, has gained recognition as a modern, powerful annotation tool in the natural language processing (NLP) landscape. Designed to support all users regardless of their computer proficiency, Prodigy's flexible design and adaptive learning make it effective for handling large datasets, such as those found in healthcare. One of Prodigy's most valuable features is its integration of active learning, allowing the tool to continuously learn as the user annotates greater amounts of data (Montani & Honnibal, 2017). The ability to dynamically learn enables Prodigy to suggest annotations, streamlining the annotation process and improving accuracy with each iteration (*Prodigy*, 2024). Prodigy's active learning functionality and ability to work with large datasets make it a valuable tool in NLP pipelines.

Prodigy was designed with usability in mind. It utilizes the popular NLP library spaCy, which powers its active learning features. As users annotate more documents, spaCy recognizes patterns in the data and fine-tunes its model, enabling Prodigy to make annotation suggestions. Like its predecessor BRAT, Prodigy supports a variety of annotation types, including entity recognition, text classification, and relationship extraction, all of which are essential when working with clinical data (Montani & Honnibal, 2017). Prodigy runs on any machine with a modern version of Python and requires no special installations or server configurations. It supports multiple input formats, including plain text, JSONL, and CSV which allows for easy integration with existing pathology text data (García Barragán, 2023).

Prodigy does have limitations that may impact its usability. Unlike many other annotation tools, Prodigy is not open-source and requires a one-time paid user license fee of $390 USD (*Prodigy*, 2024). Prodigy will provide a free license for researchers and research projects at academic institutions making it a suitable fit for the current project. The paid nature of Prodigy can pose a barrier for smaller startups, or researchers with limited budgets. Additionally, without altering the software, Prodigy runs entirely locally on the user's machine which ensures it can be used offline. However, this also presents a risk if the device is lost or damaged. This local setup may not be ideal for users or organizations requiring secure cloud storage with automatic data backup, as offered by tools like SparkNLP's Generative AI Lab ("Generative AI Lab," n.d.). Prodigy can be deployed to a cloud-based web server, but this requires users to have the know-how to deploy and manage the server (*Prodigy*, 2024).

Despite these limitations, studies and applications for annotating electronic health records (EHRs) have demonstrated Prodigy's effectiveness (Turner et al., 2022). Its active learning-based approach has been particularly helpful in healthcare projects, allowing users to quicky annotate large datasets while refining and improving the model. Prodigy has proven to be particularly useful for specific annotation tasks, such as annotating ophthalmic disease diagnoses within EHRs to assist in creating a disease registry (Macri et al., 2023). This adaptability makes Prodigy a practical choice and an investment that grows more valuable with use, providing increasingly accurate annotations over time.

Prodigy's active learning and spaCy integration make it an adaptable and efficient annotation tool for healthcare NLP. While the requirement for a paid license may limit Prodigy's accessibility to some users, its strengths in annotation efficiency, scalability, and adaptability demonstrate its value as an important annotation tool. For healthcare NLP teams aiming to optimize their annotation workflows, Prodigy represents a compelling option that combines machine learning capabilities with a practical, easy to use annotation tool.

SparkNLP (Generative AI Lab Annotations): Developed by John Snow Labs, SparkNLP is one of the most advanced NLP libraries available today and is particularly well-suited for healthcare applications. Built as an open-source library on top of Apache Spark, SparkNLP utilizes pre-trained pipelines, allowing it to annotate large amounts of data in short time (Kocaman & Talby, 2021). SparkNLP can be run in multiple programming languages, including Python, Scala, and Java.

One of SparkNLP's key advantages is its access to several pre-trained healthcare pipelines, such as HealthcareNLP. However, HealthcareNLP carries a subscription cost of $47.90/hour, which can make it expensive and less practical (*AWS Marketplace*, n.d.). SparkNLP is particularly beneficial for users needing to annotate large volumes of documents quickly, as it performs fully automated annotation through pre-trained models upon execution of the code (Kocaman & Talby, 2021).

SparkNLP has a free version but with notable limitations. It does not allow users to interact with or manually annotate reports, as it is strictly an automated NLP tool. This presents challenges for users who need to ensure each document is annotated correctly. SparkNLP requires coding in Python or Java which may be difficult for those without prior experience. To address these limitations, John Snow Labs offers a UI-based annotation tool, Generative AI Lab, which provides a more user-friendly interface for annotation ("Generative AI Lab," n.d.).

Generative AI Lab offers intuitive UI and AI-powered tools, such as pre-annotation and Auto NLP, to streamline the annotation process. Pre-annotation uses SparkNLP's pre-trained

NLP models to provide annotation recommendations, while Auto NLP functions similarly to Prodigy's active learning, adapting to user annotations and continuously training an NLP model to assist the user when annotating ("Generative AI Lab," n.d.). While Generative AI Lab does offer a user-friendly solution, it requires a paid license and either a subscription to a non-local server hosted on AWS or Azure, or a dedicated local server setup, both of which can be expensive. Although the base version of SparkNLP is free, access to a non-local server for hosting Generative AI Lab require a subscription, typically costing around $0.07/hour, with additional fees of $0.13/GB for extra processing (*Azure Marketplace*, n.d.). Access to Generative AI Lab requires a license with an associated fee. Exact costs are not disclosed but can be provided upon request ("Generative AI Lab," n.d.).

For this project, SparkNLP has notable limitations such as its reliance on automated annotation which restricts the ability to manually annotate pathology reports or quality-check the reports being processed. Another limitation is many of its useful pre-trained pipelines, such as HealthcareNLP, are behind a costly paywall (*AWS Marketplace*, n.d.). While Generative AI Lab offers many features and seems well-suited for manually annotating pathology reports, its ongoing hourly subscription costs, server hosting fees, and unknown licensing fees make it significantly more expensive than the easily accessible alternatives like Prodigy or BRAT.

SparkNLP offers a powerful, open-source NLP library, well suited for healthcare applications due to its pre-trained pipelines and automated annotation capabilities. While it can efficiently annotate large amounts of data, its reliance on automated annotation limits manual interaction presenting a challenge for users who want greater control over this process. Additionally, its more advanced features, like HealthcareNLP and Generative AI Lab, are behind costly paywalls, making them impractical for projects with constrained budgets. Overall, while

SparkNLP offers useful tools, its cost and accessibility limitations may make alternatives like Prodigy or BRAT more practical for some users.

Other Annotation Tools: There are other open-source tools that offer similar functionality but are less known due to their more recent development or limited documented use cases. This section will explore two of them: Doccano and Anafora.

Doccano, released in 2018, is a modern, open-source, UI-based annotation tool that runs locally on the user's machine. It allows users to define and color-code entities for annotation but lacks the ability to define relationships between entities (*Doccano*, n.d.). Doccano supports text and JSONL input, with JSONL as its only output format. Notably, it has been used in some medical annotation projects, such as creating an open-source annotated glaucoma medication dataset (J. S. Chen et al., 2022). A key feature of Doccano is its support for team-based annotation, making it well-suited for collaborative projects. While Doccano is usable for the ongoing project, its limited annotation features make it less suited for the requirements of this project.

Anafora, developed by researchers at the University of Colorado, is an open-source, web-based annotation tool. Designed to address the limitation of locally saved annotations, Anafora allows multiple annotators to access data remotely from a single instance running on a remote server (W.-T. Chen & Styler, 2013). As an open-source project, Anafora is regularly updated with new features and improvements. It enables users to define custom annotation schemes and label text with entities, such as parts of speech. Anafora accepts text files as input and can export annotations in various formats, including JSON. A standout feature is its support for collaborative annotation, making it a valuable asset for researchers working collaboratively on large-scale datasets and annotations (W.-T. Chen & Styler, 2013). Anafora is a useful annotation tool, but with limited research into the software and the lack of need for collaborative features in this project, it is less desirable for this project.

Label-Studio, released in 2020, is an open-source, user-friendly annotation tool that runs locally on the user's machine using Python. It supports the annotation of various data types, including images, video, audio, and text (Tkachenko, 2020). Entities and relationships for annotation can be easily defined either through the program or by importing an XML file. Label-Studio offers flexibility with a wide range of import and export formats, including text, audio (WAV), and JSON, along with many output options. The tool is easy to set up, requiring only a single bash command to get started, though account creation is necessary to access the annotation features. Additionally, the platform integrates with third-party AI models to assist with annotation if needed. While Label-Studio is well-suited for the current project, its requirement for account creation and its extensive range of annotation types makes it less ideal, as this project only requires text annotation.

2.3 Conclusion

Modern annotation tools have evolved to meet the needs of researchers and healthcare professionals. In finding the balance between accessibility, scalability and automation, BRAT, Prodigy, and SparkNLP each bring unique advantages to the annotation field. Despite its outdated interface and limited support, BRAT remains a valuable tool due to its user-friendly design and open-source nature, making it widely available. It is particularly useful for smaller projects needing the ability to define entities and relationships when annotating. Prodigy offers a more modern approach to annotation, featuring active learning and spaCy integration, making it ideal for projects with larger datasets, particularly in healthcare. Its primary drawback is its cost which may limit access for users or projects with limited funding. SparkNLP, with its automated, large-scale annotation capabilities and specialized pre-trained models, offers a more efficient solution for processing large datasets. However, full access to its functionality, including its UI-based annotation tools and pre-trained pipelines, requires a substantial financial commitment due to licensing and ongoing subscription and usage fees. The substantial cost can be the limiting factor for projects with budget constraints when deciding on annotation software. Additionally, SparkNLP's automated annotation approach may be less than ideal for users who need control over their annotations where manual reviews and precision may be necessary for certain projects. Other open-source annotation tools, such as Doccano and Anafora, offer useful features like collaborative functionality and unique user interfaces. However, their limited features and the lack of research on each tool make it difficult to determine if they are a good fit for this project.

This review finds Prodigy to be an ideal middle ground for annotation, providing valuable functionality alongside its free access for researchers, making it ideal for this project. Its integration with spaCy also enhances annotation efficiency over time. Each annotation tool addresses different aspects of the evolving needs in NLP, from ease of use and adaptability to machine learning integration. As the field advances, the choice of annotation software will depend on factors such as project scale, budget, and automation requirements. While open-source tools like BRAT underscore the importance of accessibility and collaboration, Prodigy's features and specific use cases position it as the top choice for annotating medical pathology reports.

Methodology

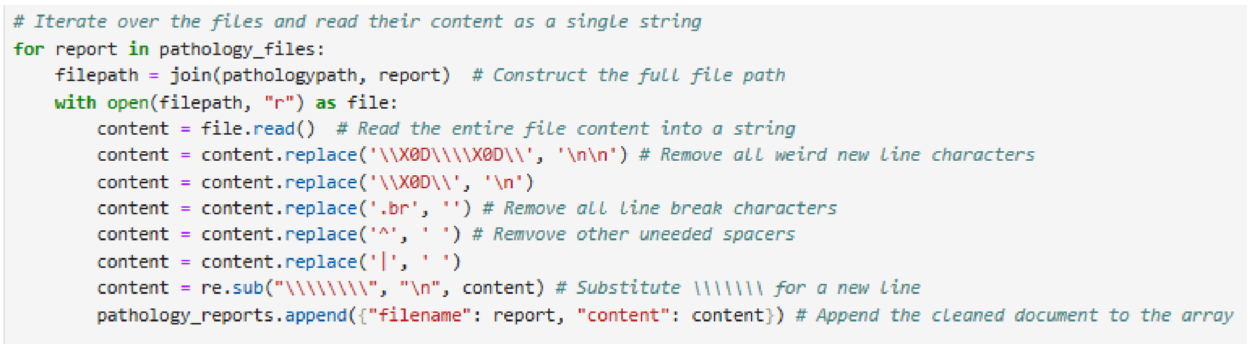
3.1 Introduction

This section outlines the methodology used to create an annotated dataset of breast cancer pathology reports. The process involved evaluating annotation tools, sorting pathology reports by cancer type, manually annotating selected reports, and converting and standardizing the data. Additionally, this study assesses the effectiveness of three annotation tools: Prodigy, Doccano, and Label-Studio, to determine their suitability for annotating cancer pathology reports. Multiple annotation tools were selected for this project to identify the best tool for future work. Additionally, the conversion code developed in this project ensures that future annotations, regardless of the tool used, can be included in the annotated dataset for NLP model training.

An annotation guideline was created as to limit inconsistencies across reports. Once a subset of reports was annotated using each tool, the output data was then converted into a standardized format to support NLP model training and ensure compatibility with all three annotation tools. The methods described in this section establish the foundational steps necessary for developing an NLP model and document the approach taken in this project.

3.2 Sorting Pathology Reports by Cancer Type

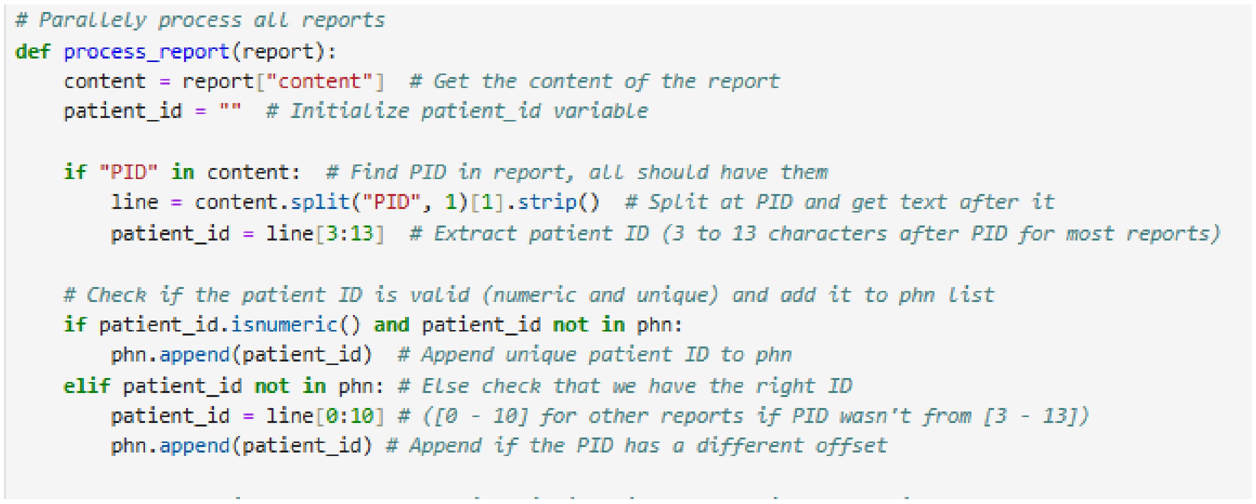
The first outlined goal of this project was to sort 18,488 cancer pathology reports from 7,519 patients by their diagnosed cancer type as found in their pathology reports. This dataset is from BC Cancer and consists of patients who have had a recurrence of cancer. Before starting the sorting process, the text within the pathology reports needed to be cleaned to allow for easier readability during the annotation process. In raw text form, likely originating from the DICOM format, the reports contained numerous unnecessary characters, such as \\X0D\\ tags indicating new lines, as well as .br tags and vertical bars (|) used in place of spaces. The unnecessary characters were removed using both the .replace() and .sub() functions, as shown in *Figure 1* below.



```
# Iterate over the files and read their content as a single string
for report in pathology_files:
    filepath = join(pathologypath, report)  # Construct the full file path
    with open(filepath, "r") as file:
        content = file.read()  # Read the entire file content into a string
        content = content.replace('\\X0D\\\\X0D\\', '\n\n') # Remove all weird new line characters
        content = content.replace('\\X0D\\', '\n')
        content = content.replace('.br', '') # Remove all line break characters
        content = content.replace('^', ' ') # Remvove other uneeded spacers
        content = content.replace('|', ' ')
        content = re.sub("\\\\\\\\", "\n", content) # Substitute \\\\\\\ for a new line
        pathology_reports.append({"filename": report, "content": content}) # Append the cleaned document to the array
```

Figure 1: Code for Cleaning Pathology Reports

Once cleaned, the reports became significantly easier to read. Next, the focus shifted to sorting the pathology reports by each patient's Personal Health Number (PHN). Despite the differences across reports, all of them included the patient's name, the pathologist, and the patient's PHN within the DICOM header data. This made it much easier to match pathology reports to the corresponding patients. Within the header data, the patient's PHN always appeared in one of two locations, following the 'PID' string, which denotes Patient ID. The .split() function was used to locate the first occurrence of 'PID' in each report. Since PHNs are always 10 characters long, they could be reliably extracted by specifying either the 3rd to 13th or 0th to 10th characters following the matched 'PID' string, as shown in *Figure 2* below. Once the PHNs were identified, each report was saved in a new folder named after the corresponding PHN. This process organized all reports by the patient's PHN, allowing for the grouping and analysis of pathology reports to identify the cancer type for each patient.



```
# Parallely process all reports
def process_report(report):
    content = report["content"]  # Get the content of the report
    patient_id = ""  # Initialize patient_id variable

    if "PID" in content:  # Find PID in report, all should have them
        line = content.split("PID", 1)[1].strip()  # Split at PID and get text after it
        patient_id = line[3:13]  # Extract patient ID (3 to 13 characters after PID for most reports)

    # Check if the patient ID is valid (numeric and unique) and add it to phn list
    if patient_id.isnumeric() and patient_id not in phn:
        phn.append(patient_id)  # Append unique patient ID to phn
    elif patient_id not in phn: # Else check that we have the right ID
        patient_id = line[0:10] # ([0 - 10] for other reports if PID wasn't from [3 - 13])
        phn.append(patient_id) # Append if the PID has a different offset
```

Figure 2: Code to Retrieve PHN for Sorting Pathology Reports

With the reports now cleaned and sorted by PHN, the focus shifted to sorting the pathology reports by the patient's diagnosed cancer type. This was achieved through keyword matching. A set of unique identifying cancer keywords were chosen to represent each type of cancer. These keywords were identified through trial and error by extracting terms from the reports and analyzing which ones yielded the best results. To achieve this, all pathology reports for a single patient were loaded into an array, and the occurrence of each relevant keyword for a specific cancer type was counted and tracked. After analyzing all reports from a single patient, a list was returned showing the counts of each keyword found, organized by cancer type. This gives us a strong indication of what type of cancer the patient has. This was done for all patients and pathology reports. If a single cancer type had the most keywords identified for a patient, it would be considered a positive diagnosis, and the reports would be saved in the corresponding folder. For example, if the breast cancer keywords 'HER2,' 'Breast Cancer,' etc., were found twelve times in a patient's reports, while other cancer keywords appeared only once or twice, the patient would be considered to have breast cancer, and their reports would be saved under the 'Breast Cancer' folder. If a conflict arose where two cancer types have the same number of occurrences, such as both skin and breast cancer having two occurrences each, the pathology report would be re-reviewed for negation keywords, such as 'not' or 'none.' If any of these negation keywords are found within 150 characters of a cancer keyword, the keyword would then be excluded from the count. This was done to improve sorting accuracy and minimize conflicting reports. After checking for negation keywords, the reports would be saved in their respective cancer folders based on the cancer type with the most keywords. This resulted in 17,894 of the 18,488 cancer pathology reports being classified by cancer type, with only 714 reports unable to be sorted into their correct folder and thus stored in a 'Conflicting Type' folder.

The algorithm for sorting pathology reports by cancer type does have some limitations. For example, it is not guaranteed that the sorted cancer type will be correct due to the possibility that keywords can be found within the report, that are not related to the patient’s current cancer type. Negation checking is not entirely reliable, as medical professionals may refer to other items within the 150 characters of a keyword. However, since it is only used in cases of conflicting cancer types, it does somewhat improve the overall reliability of the sorting process. The absence of a written cancer diagnosis in each report means that, without the assistance of a medical professional or AI model, we cannot guarantee the cancer diagnosis for each patient. This is due to variations in pathology reports, missing or conflicting diagnostic information, and the lack of a stated diagnosis in many reports. Without the assistance of a medical professional to help in analyzing each report that has been sorted, we cannot guarantee this algorithm won’t provide any false positives. Some of the sorted pathology reports were reviewed with a second-year medical student and found to be correctly sorted, but this was limited to less than 100 pathology reports. An NLP model was not initially used to sort the pathology reports due to uncertainty about whether it was allowed, given the sensitive nature of the reports. This issue has since been resolved.

This initial sorting process resulted in

- 287 Bladder Cancer Pathology Reports
- 224 Brain Cancer Pathology Reports
- 7,723 Breast Cancer Pathology Reports
- 191 Cervical Cancer Pathology Reports
- 825 Colon Cancer Pathology Reports
- 197 Kidney Cancer Pathology Reports
- 685 Leukemia Cancer Pathology Reports
- 193 Liver Cancer Pathology Reports
- 2,668 Lung Cancer Pathology Reports
- 854 Lymph Cancer Pathology Reports
- 97 Pancreatic Cancer Pathology Reports
- 1,028 Skin Cancer Pathology Reports
- 67 Stomach Cancer Pathology Reports
- 336 Thyroid Cancer Pathology Reports
- 2,157 Ovarian Cancer Pathology Reports
- 205 Vulvar Cancer Pathology Reports
- 751 Cancer Pathology Reports that could not be sorted

If time permits, a second version of the sorting algorithm will be tested using the NLP model medspaCy to assess whether AI can improve the accuracy of the sorted cancer pathology reports. The model must be run locally due to the sensitive nature of the pathology reports.

Future work can improve the accuracy of this sorting process, but for now, the initial sorting method has proven reliable enough to continue with the project.

3.3 Comparison of Annotation Tools

Of the six annotation tools reviewed in the literature, only **Prodigy**, **Doccano**, and **Label-Studio** were found to be suitable for this project, as all three were easy to install on PHSA machines and proved effective for annotating breast cancer pathology reports. In contrast, tools such as Anafora and SparkNLP require access to cloud servers to store the annotations, which do not comply with PHSA privacy regulations and as such they were not able to be used for this project.

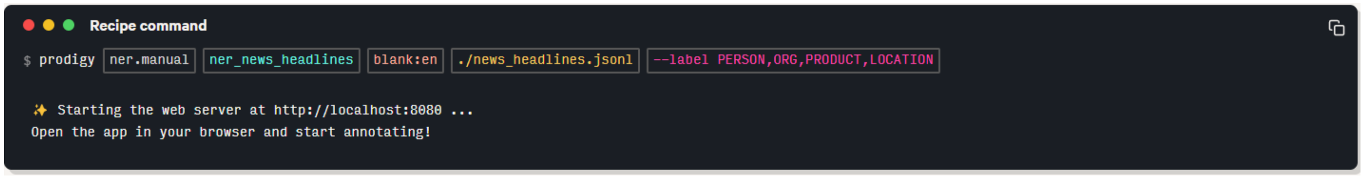
3.3.1 Annotation Tools Not Used

Several annotation tools were considered but ultimately not selected for this project due to various limitations:

- **BRAT**: This tool was excluded due to its reliance on outdated software (Python 2.5), an outdated user interface, and the requirement to run on a Linux machine. BRAT was tested locally via WSL (Windows Subsystem for Linux), the setup proved challenging, and the tool was unintuitive for annotation.
- **SparkNLP**: This tool was not selected due to its high cost and reliance on cloud services such as AWS or Azure, which are unsuitable for the sensitive nature of the pathology reports. For these reasons, SparkNLP is impractical for this project.
- **Anafora**: Anafora was also excluded because it required a Linux server to host the collaborative annotation tool, which also proved unsuitable for the sensitive nature of the cancer pathology reports and as such was not chosen for this project.

3.3.2 Annotation Tools used

Prodigy: Requires a license to begin annotating but, but with a valid post-secondary email, a free license can be obtained by contacting the developers of Prodigy. Prodigy can be run on any Windows machine with an up-to-date version of Python 3.0. Setting up Prodigy proved to be relatively difficult as Prodigy does not provide a UI (User Interface) to either upload the documents or create the annotation entities you will be working with. To set up Prodigy as shown in *Figure 3*, all required items must be specified through a bash command. These include the annotation recipe (e.g., *ner.manual* for manual annotation), the dataset name (e.g., *ner_news_headlines*), the spaCy model to be used (*blank:en* indicates no model), the file to annotate (e.g., *./news_headlines.jsonl*), and the labels to be used specified with the --label flag.

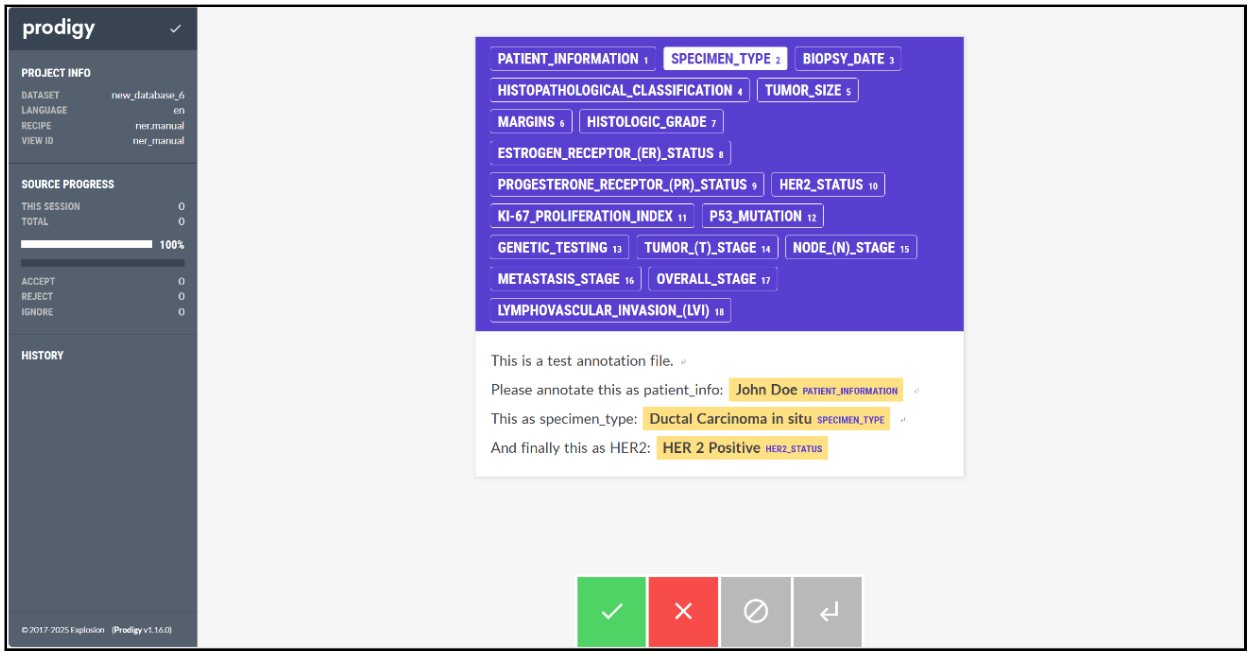


```
$ prodigy ner.manual ner_news_headlines blank:en ./news_headlines.jsonl --label PERSON,ORG,PRODUCT,LOCATION

 ✨ Starting the web server at http://localhost:8080 ...
 Open the app in your browser and start annotating!
```

Figure 3: Bash Command for Starting Prodigy. (Prodigy, 2024)
Manually annotating new_headlines.jsonl with the labels: Person, Org, Product and Location

Prodigy's documentation provides instructions for loading the correct documents and labels for annotation, but it still requires some knowledge of bash to get working. Prodigy can accept text or .csv files, but it splits each line into a new page when loading these file types. To effectively annotate the documents, files should be converted into the JSONL format. For this project, this conversion was done in Python. This conversion resulted in some formatting issues such as missing new lines, but this did not prove to be an issue as no important data was lost. Once started, annotations in Prodigy were done by selecting an entity and highlighting the text you would like to annotate. The highlighted text is shown with a yellow fill around the text and the entity name at the end of the highlight, this is shown in *Figure 4* below.



DATASET new_database_6
LANGUAGE en
RECIPE ner.manual
VIEW ID ner_manual

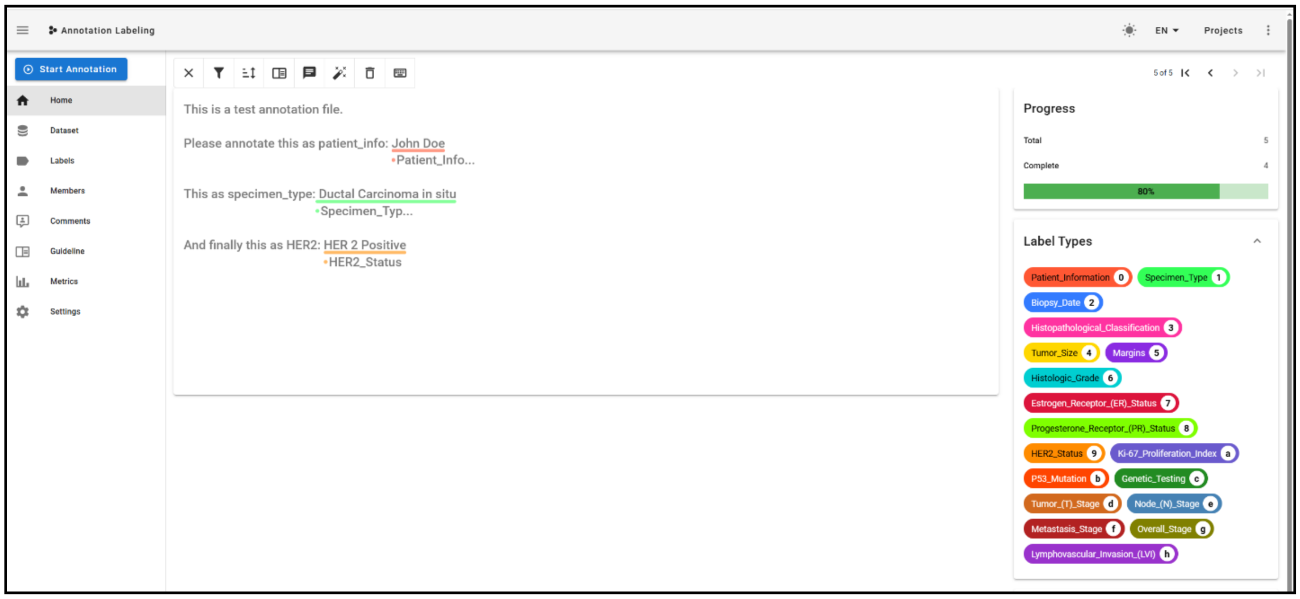
THIS SESSION 0
TOTAL 0
100%
ACCEPT 0
REJECT 0
IGNORE 0

This is a test annotation file.
Please annotate this as patient_info: John Doe PATIENT_INFORMATION
This as specimen_type: Ductal Carcinoma in situ SPECIMEN_TYPE
And finally this as HER2: HER 2 Positive HER2_STATUS

Figure 4: Prodigy User Interface for Annotation

Deletions were made in Prodigy by selecting the annotation you would like to remove and clicking the 'X' button in the top left corner of the annotation. The green checkmark and red 'X' at the bottom of the page are used to accept or reject all annotations within the current page. Clicking the checkmark saves the annotations and advances to the next page, while clicking the red 'X' discards the annotations. Once annotations are complete, the annotations must be saved to a database. This is done by selecting the save icon in the top left corner beside the Prodigy logo (once saved it becomes a checkmark). To write the annotated data from the Prodigy database to a file, additional bash commands are required. You need to specify the local Prodigy database where the annotations are saved and provide a file location for the output. This results in an annotated JSONL file containing all annotated Prodigy data. The JSONL file includes a large amount of data, with pre-tokenized text, annotations, and the required metadata. It is important to note that a single Prodigy database stores all annotation data, not just data for individual reports. Prodigy proved to be a very helpful annotation tool. Its easy-to-use and straightforward UI made annotating data quick and seamless, with no issues encountered during the annotation process. The ability to double-click text for automatic highlighting was especially useful. However, Prodigy can be slow to set up and navigate between files, as the server must be stopped, and a new command entered each time a different file is to be annotated.

Doccano: Is an open-source annotation software with a simple setup and installation process. It requires Python 3.0 or newer to run and can be installed with bash commands found in the Doccano repository (*Doccano*, n.d.). To initialize Doccano, some bash commands are required. In one terminal, use *doccano init* to start the server, followed by *doccano webserver --port 8000* to specify the localhost location. Then, in a separate terminal, use *doccano task* to begin the program. Once Doccano starts, a new sequence labeling project must be created and given a name. Doccano supports different labeling types, but the main one required for this project is sequence labeling. Afterward, you can import the data you wish to annotate under the 'Dataset' tab by selecting 'Import' and choosing the files you wish to annotate. Multiple files and types can be selected and uploaded at once, as Doccano supports both text and JSONL files, either of the formats worked equally well in Doccano. Then, under the 'Labels' tab, you can create and add the entities you wish to use. Each entity can also be assigned a color. To begin with annotating, once the correct file and entities are loaded, navigate to the 'Home' tab and select 'Annotate' on the file you wish to begin annotating. Select the label you would like to use from under 'Label Types' on the right-hand side of the screen. Then, highlight the text using the mouse. The selected text will be labeled with the chosen entity and highlighted with the corresponding color. This is shown in *Figure 5* below.


This is a test annotation file.

Please annotate this as patient_info: John Doe
Patient_Info...

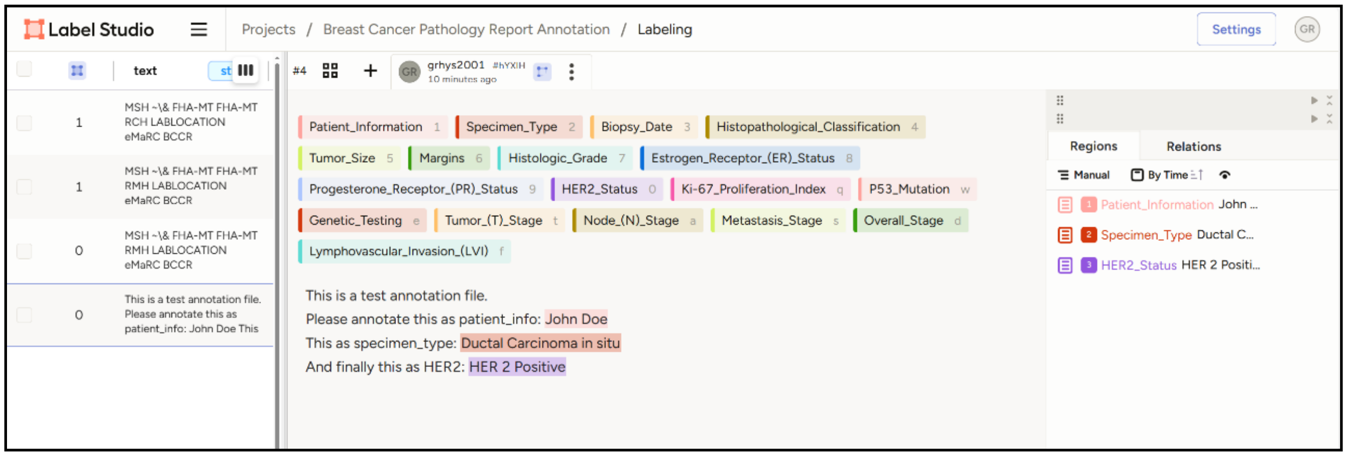
This as specimen_type: Ductal Carcinoma in situ
Specimen_Typ...

And finally this as HER2: HER 2 Positive
HER2_Status

Figure 5: Doccano User Interface for Annotation

To remove an annotation, select the annotation which opens a drop-down list. From there, you can view the current label and remove it by selecting the 'X' at the top of the drop-down. Then, once you have finished annotating, navigate back to the 'Dataset' tab where files can be marked as complete. Then once marked as complete, select the 'Export' button and the annotated data will be downloaded as a JSONL file. Doccano was simple and quick to install, with a smooth and user-friendly setup process. Uploading files and adding entities was fast and intuitive. However, the actual annotation process was slower compared to Prodigy.

Label-Studio: Label-Studio is another open-source annotation tool that can be used to annotate a range of items from videos to images to text. Similarly to the previously mentioned tools, Label-Studio requires Python 3.0 to install and run locally on a Windows machine. To begin annotating, type *label-studio* into the terminal and it will automatically start a locally hosted server to begin annotation. First, log in, then create a new project and assign it a name. Then navigate to the 'Data Import' tab and upload the documents you wish to annotate, though Label-Studio accepts all document types, it's best to annotate documents stored in a JSON format. Next, under the 'Labeling Setup' tab, set the project type to 'Text Classification'. Labels can then be added manually or uploaded via an XML file that includes both the label names and their associated colors. Once on the home page, each file can be annotated individually. Select the file you wish to annotate, and it will open the annotation page, as shown in *Figure 6* below.



	text
1	MSH ~\& FHA-MT FHA-MT RCH LABLOCATION eMaRC BCCR
1	MSH ~\& FHA-MT FHA-MT RMH LABLOCATION eMaRC BCCR
0	MSH ~\& FHA-MT FHA-MT RMH LABLOCATION eMaRC BCCR
0	This is a test annotation file. Please annotate this as patient_info: John Doe This

This is a test annotation file.
Please annotate this as patient_info: John Doe
This as specimen_type: Ductal Carcinoma in situ
And finally this as HER2: HER 2 Positive

1 Patient_Information John ...
2 Specimen_Type Ductal C...
3 HER2_Status HER 2 Positi...

Figure 6: Label-Studio User Interface for Annotation

Your labels will appear at the top of each document and will have shortcut for each label you included (typically 0 – 9 then a – z) to enable that label. Then to highlight the text, select the label you would like to work with, highlight the text as you typically would in most programs using the mouse. To change the label, you can enable a different one and re-highlight the same text. To delete a label, a table on the right side of the screen displays each labeled entity in the order they were highlighted. Each entity will have a trashcan icon that can be selected to remove that specific label. Each of these entities within the table can also be selected to display that label on the page. Once annotation is complete, to save a document, navigate back to the project home page and select export, this will give you a list of file types to select. Select your desired file type and 'Export' and your annotated data will be downloaded. For this project, a JSON file type was selected to keep consistent with the JSONL formats of the other two annotation tools. Label-Studio was similar to Doccano, with an easy setup and a functional UI for uploading documents and adding labels. However, the annotation process was again slower compared to Prodigy.

3.4 Manual Annotation of Reports

Before beginning the annotation process, labels were created to identify key information in the pathology reports. A second-year medical student developed an annotation guideline to ensure consistency across the reports. The annotation guidelines were refined over several weeks during the initial annotation process to identify the most appropriate labels for the pathology reports. The table on the next page (*Figure 7*) outlines the guideline, detailing the annotation categories, descriptions, and text examples. The annotation process was done collaboratively with the medical student for their expertise in identifying pathological information. Throughout the process, ambiguous text was consistently recognized. If the intended meaning of a medical professional’s pathology report text was unclear, the corresponding items were left uncategorized due to the difficulty in interpreting their meaning. These annotations help capture important information for the NLP model to extract from the cancer pathology reports.

Categories	**Details**	**Examples**
Patient Information	Includes Name, Patient ID (MSP/PHN), age/DOB, sex, physicians All names highlighted	• AGE/SX 67yrs/F • Dr. Lisa Cuddy • DOB: 1957/07/31 • PHN: 900011122
Specimen Type	Specifies the type of biopsy	• Left axilla • Left breast • Breast Bx @ 3 o'clock
Biopsy Date	Date of biopsy; for tumors and lymph nodes	• Extracted June 23, 2025, at 1500
Histopathological Classification	Indicate tumor type (e.g., invasive ductal carcinoma, DCIS).	• DCIS • In-situ invasive carcinoma
Tumor Size	Size of actual tumor	• 2.4 mm at largest dimension
Margins	Margins: distance given for each margin space	• Lateral margin: clear • Medial margin: clear • Invasive margins
Histologic Grade	Nottingham grade given after cancer cells are analyzed	• Overall Nottingham grade • Histologic grade ⅔ (glands, nuclei, mitoses)
Estrogen Receptor Status	Positive or negative, with an allred score	• ER: Positive (Intensity: Strong, 67-100%), All red score: 8/8
Progesterone Receptor Status	Positive or negative, with an allred score	• PR: Positive (Intensity: Strong, 67-100%), All red score: 8/8
HER2_Status	HER2 Status, usually Positive or Negative	• HER2 (+)
Genetic Testing	All biomarkers found from testing	• Pancytokeratin AE1/AE3 • Cytokeratin 34BE12 • E-cadherin
Node Stage	Cancer stage within the lymph nodes	• Lymph nodes negative for metastases
Metastasis Stage	Stage of cancer, whether it has metastasized	• Micro/macro metastasis not found
Overall Stage	Other staging, not under node or metastasis	• Pathological stage pT2pN0
Lymphovascular Invasion	Cancer is present in the lymphatic cells, usually identified or not identified	• Lymphatic or vascular invasion • Lymphovascular invasion: Grade 2

Figure 7: Table of the Annotation Guidelines Followed

With assistance from the second-year medical student, a total of 43 pathology reports were annotated across the three tools:

- **Prodigy**: 22 reports (17 unique, 5 duplicates)
- **Doccano**: 11 reports (6 unique, 5 duplicates)
- **Label-Studio**: 10 reports (5 unique, 5 duplicates)

Including both unique and duplicate reports ensures that the conversion software works correctly and can handle slight variations in the data across annotations.

3.5 Data Conversion between Annotation Tools

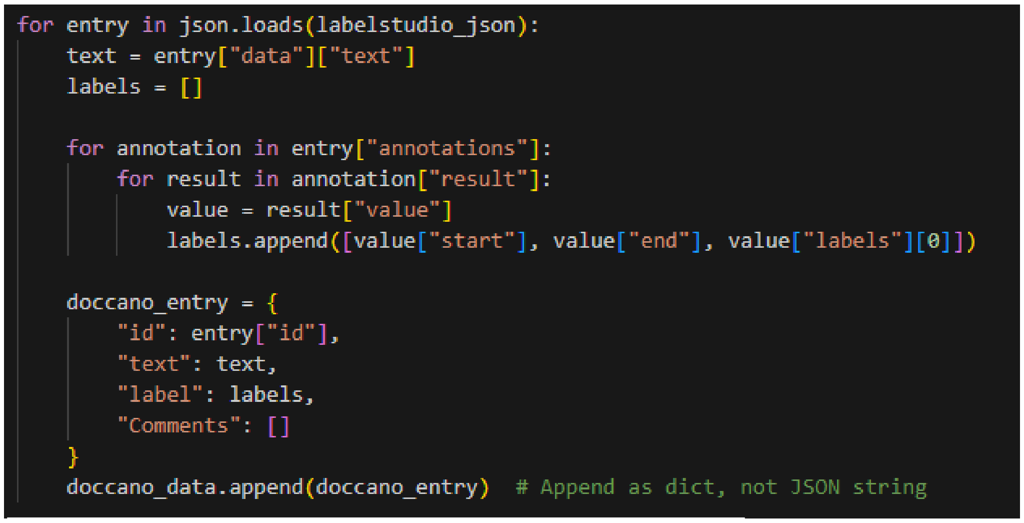
The conversion process between the three annotation tools involves running six separate programs to ensure the output from each tool can be converted into a format that is compatible with the other two tools. The initial structure of the code is similar for each tool, it loops through each directory containing the annotated reports, identifies files with extensions .json (Label-Studio) or .jsonl (Prodigy and Doccano), and loads each file into a separate array in Python. From this point, the conversion function is called.

3.5.1 Converting to Prodigy

Converting from the output of either file type to Prodigy requires some imports not otherwise seen. It loads the English tokenizer (en_core_web_sm) from spaCy, which breaks the text into individual tokens such as words and punctuation. Tokenization is essential because Prodigy works with tokens rather than individual words. To ensure the conversion is compatible with Prodigy, a hash function is used to create a unique identifier for both the input and output text. Next, we find the span or distance between the tokens where an annotation is. This is required because annotated data in Prodigy uses both the starting and ending characters as well as the starting and ending tokens to represent an annotated item. Finding both tokens works by matching the input annotations label with a start and end position, then searching through the tokenized text to identify which tokens best correspond to that span. It first attempts exact matching by checking if a token's start and end indices overlap with the character span. If that fails, it looks for overlapping tokens and adjusts the span accordingly. This gives us all required data needed to convert the annotations to Prodigy, then we can recreate a .jsonl file with all the important data such as the text, tokens and annotator id. Once all required metadata is included, the data is now converted to Prodigy and can be written to a file. The inputs given by Doccano and Label-Studio were somewhat similar, with the important fields being the 'text' and 'labels' keys from Doccano and the 'text' and 'value' key from Label-Studio. Both keys provide the same data, the annotated data with a start and end character under 'labels' or 'value' as well as the raw text under 'text'.

3.5.2 Converting to Doccano

Doccano conversions were relatively straightforward, as the output data only requires three key items: the 'text' field, which contains all the text from the original file; the 'labels' field, which includes the entity type along with the start and end character positions to indicate what has been annotated; and a 'comments' field to store any additional comments the user may have added. When converting annotations from either Prodigy or Label-Studio to Doccano, the essential task is to extract the label type (e.g., 'HER2') along with the start and end character positions of the annotation. In Prodigy, the annotation type is contained within the 'span' field, while in Label-Studio, it is referred to as 'value'. These annotations were then converted into the Doccano format, which requires the start and end positions along with the label. Once the script loops through all items in the input data, it can reconstruct the JSONL with the appropriate text, labels, and any other necessary fields the code for this is seen in *Figure 8* below. For Prodigy data, an offset may be required if the annotations span multiple lines in the JSONL file, ensuring that the character positions align correctly in the final output.



```python
for entry in json.loads(labelstudio_json):
    text = entry["data"]["text"]
    labels = []

    for annotation in entry["annotations"]:
        for result in annotation["result"]:
            value = result["value"]
            labels.append([value["start"], value["end"], value["labels"][0]])

    doccano_entry = {
        "id": entry["id"],
        "text": text,
        "label": labels,
        "Comments": []
    }
    doccano_data.append(doccano_entry)  # Append as dict, not JSON string
```

Figure 8: Code for converting from Label-Studio to Doccano

3.5.3 Converting to Label-Studio

Converting annotations from Prodigy or Doccano to Label-Studio requires reformatting the JSONL output data to match Label-Studio's structure. Prodigy stores annotations in the 'spans' field, which includes the start and end positions along with the label, while Doccano uses a 'label' field that holds the annotation data. The conversion begins by extracting the relevant data (starting character, ending character, and the label) from either field. For Prodigy data, offsets are applied to account for the annotations that span multiple lines in the JSONL file, ensuring the character positions align correctly with the full text. After loading and extracting the annotation data, it is converted into a Label-Studio compatible format, with each annotation represented as a 'value' object containing the label, start and end positions, and the corresponding text. After this, unique IDs were generated for each document, and necessary metadata is added to the JSON format to be compatible with Label-Studio such as timestamps and details about the project are added. Since a creation timestamp cannot be gathered from either Prodigy or Doccano, the timestamp is generated during the runtime of the program. The processed data is then combined into a JSON with the entries Label-Studio requires including the original text, annotations, and metadata. The conversion process ensures all annotations were converted correctly from either software to Label-Studio.

3.6 Conclusion

The methodology for this project was designed to ensure effective sorting, annotating and annotation conversion when working with cancer pathology reports. The process involved sorting pathology reports by cancer type using keyword matching, followed by manual annotation with the support of various annotation tools, Prodigy, Doccano, and Label-Studio. These tools were evaluated based on functionality, ease of setup, and compatibility, and all were found to be suitable for the needs of this project. The annotations were standardized and converted into compatible formats to support future training of an NLP model. Challenges, such as the limitations of the sorting algorithm and discrepancies in the annotation process, were addressed through iterative refinement, ensuring reliable results. The project also demonstrated the importance of collaborating with a medical student, whose clinical knowledge helped support the annotation process and improve understanding of the pathology reports. Overall, the methodology outlined in this section sets the foundation for developing an NLP model through the sorting, annotating and conversion of cancer pathology reports with future work aimed at continuing the annotation of pathology reports and training of an NLP model.

Results

Due to the nature of this project, traditional quantitative or qualitative analyses cannot be applied. The annotation process does not give any numerical data that is suitable for statistical analysis, nor does it provide us with any measures of annotation accuracy. Additionally, assessing the effectiveness of the sorting algorithm remains challenging, as pathology reports do not explicitly include cancer classification. Instead, classification must be inferred from medical terminology within the reports, a process that proved difficult without a specialized medical background.

Annotation Progress

The annotation process resulted in the following:

- A total of **43** Breast Cancer Pathology Reports were annotated.
- **33** unique Breast Cancer Pathology Reports were included in the dataset.

Annotation Observations

Throughout the annotation process, reports with structured data and clearly labelled headers such as Lymphovascular Invasion, Margins and ER/PR status proved easier to annotate. This is compared to the pathology reports with important information scattered throughout the report in various comments. Pathologists often copied and pasted details from previous reports into new ones for the same patient, resulting in lengthy pathology reports with repeated text across multiple documents. This would slow down the annotation process, from an average time of approximately 10 minutes per report to closer to 20 minutes per report for the longer reports. Ambiguous text also led to the slowing of the annotation process, as the medical professionals would often use shorthand or terms not everyone is familiar with when writing the report. Items such (Bx) referring to a biopsy or (hot not blue) referring to lymph nodes with a high radioactive count required some research to understand and correctly annotate these items. The time constraints of this project resulted in fewer pathology reports annotated than originally planned. The goal of the project was to annotate 75 Breast Cancer Pathology reports (25 in each tool) but due to limitations such as time constraints required to annotate, creating as well as refining the annotation guideline and lengthy, repetitive pathology reports, only 43 Breast Cancer Pathology reports were annotated.

Many of the reports contained well defined sections where important items such as HER2 and ER/PR diagnosis could be easily found. Labels such as genetic testing and biopsy date were used sparingly as not all pathologists included this information in the reports. However, it was still important to include all labels to ensure consistency, even if certain information did not appear in every report.

Future Directions

While the findings of this project lay the foundation for annotating cancer pathology reports, future work can focus on ways to accelerate the annotation process. This can be achieved using integrated AI models, such as those available in both Prodigy and Label-Studio, which learn from the annotations and can provide suggestions during the annotation process. While this process is not without its limitations, it can significantly improve the speed of the annotation process. Future work will allow for a more accurate assessment of the performance and capabilities of the trained NLP model. For this project, we can only evaluate the progress made to date.

Discussion

This project was largely successful in meeting its goals, including the development of an annotation guideline specifically for breast cancer pathology reports. The only shortcoming was in the total number of reports annotated, only 43 out of the intended 75 were completed. Following the creation of the annotation guideline, 43 breast cancer pathology reports were annotated in accordance with it. This project also provided a comparative analysis of the three annotation tools used. The pre-processing required for annotating the reports was completed, including cleaning the pathology reports and sorting them by PHN and cancer type. This resulted in cleaner, more comprehensible reports for the annotation process. After annotating 43 breast cancer pathology reports, software was developed to convert between different annotation formats, ensuring consistency and facilitating future annotation work.

While the annotation process proved effective, several key insights and limitations emerged throughout the project.

Sorting Pathology Reports by Cancer Type

The sorting process, which relied on keyword matching and negation detection, successfully categorized 17,894 out of 18,488 reports. However, 751 reports remained uncategorized due to conflicting or insufficient keyword occurrences. While keyword-based classification provided to be a viable solution for sorting pathology reports without needing to fully comprehend the reports, its reliance on predefined terms introduced limitations. Certain pathology reports may not contain enough occurrences of the predefined terms to yield a confident match. The sorting process may also be unreliable if alternative terminologies are used, or if the report includes information about a past cancer diagnosis or metastasis to other areas, which could lead to incorrect sorting or the inability to identify the cancer type. Additionally, while negation detection helped improve sorting accuracy, its limited context window of 150 characters around the matching keyword posed constraints. This limitation could lead to errors if negation words were misinterpreted within the pathology notes, failing to capture the intended meaning of certain statements. Although negation detection was only applied in cases where conflicting cancer types were found after processing keyword detection, there remains a possibility that this process introduced errors during the sorting of the reports.

The absence of written cancer diagnoses within reports also contributed to challenges when classifying the reports. Through manual validation of a subset of reports, a low false-positive rate was found. However, full validation by a medical expert or the integration of a machine learning model could further enhance classification reliability. Future work may explore machine learning approaches such as incorporating an NLP model trained on medical data to identify patient cancer types with greater accuracy and precision.

Comparison of Annotation Tools

Through an analysis of the three annotation tools used, Prodigy, Doccano, and Label-Studio, each tool highlighted key trade-offs between usability, difficulty of setup and annotation efficiency. Each tool displayed strengths and weaknesses throughout the duration of this project:

- **Prodigy** demonstrated strong annotation capabilities but required difficult command-line interactions for setup, operation, and saving. Although this presented an initial learning curve, annotation became more efficient with continued use.
- **Doccano** offered a user-friendly web-based interface and simplified entity labeling, allowing work on multiple documents per annotation session. The setup process was straightforward, and working with documents in their default text format proved much easier. While the annotation process was slower compared to the other tools examined,

primarily due to the need to manually select entity type then highlight each word, the ease of setup and clearly defined output data made this a reasonable trade-off.

- **Label-Studio** offered versatile annotation capabilities across multiple data types. However, the annotation process was slower than expected due to the UI design, which placed entity selections out of reach when working with larger reports. Despite being the slowest tool in terms of annotation speed, the output data was easy to work with, and the tool proved effective for annotating pathology reports.

The decision to exclude BRAT, SparkNLP, and Anafora was based on practical constraints, including outdated software dependencies, high implementation costs, and requirements incompatible with the project setup. This highlights the importance of considering both feasibility and user accessibility when selecting a tool for data annotation.

Manual Annotation and Standardization

The annotation process highlighted the need for standardized guidelines to ensure consistency across annotated reports. By applying these guidelines during the annotation process, inconsistencies can be minimized, ultimately improving the quality of the data for future NLP applications. However, the manual nature of the annotation process remained a time-consuming and limitation aspect of the project. Future work could benefit from semi-automated annotation pipelines that work with pre-trained language models to suggest annotations, reducing the time spent annotating reports.

Additionally, converting the outputs of each annotation tool to ensure compatibility and standardized data helps support NLP model training by providing consistent input, regardless of what annotation tool was used. However, additional work was required to reliably convert more complex data, such as that provided by Prodigy, due to its method of tokenizing text.

Limitations and Future Directions

Despite the general success of this project, several limitations have been noted. The keyword-based sorting approach, while effective, lacks the flexibility to adapt to variations in text found within the pathology reports. Incorporating pre-existing NLP models, such as medspaCy, could significantly improve cancer classification accuracy by better understanding the context in which words are found, going beyond simple keyword counts.

The annotation process, while methodically structured, relied on human annotators and thus remained susceptible to mistakes or variations in the annotations. These issues could be addressed through a review process or by involving additional collaborators, such as medical professionals, to review and refine the annotated data, ensuring consistency across reports. However, for use in training an NLP model, the data does not need to be perfectly consistent, as slight variations are acceptable for training purposes.

Overall, this study lays the groundwork for the structured annotation of breast cancer pathology reports, while also highlighting key areas for improvement. Through the incorporation of an NLP model and utilizing experts from the field, future research can enhance the annotation process, leading to more precise and efficient medical text annotation.

Conclusion

This project provides an important step forward in advancing the field of medical informatics by laying the groundwork for training an NLP model. It involved the development of an annotation guideline and creation of a standardized annotated dataset of breast cancer pathology reports. The multi-step approach, including data cleaning, sorting by cancer type, manual annotation with a medical student providing medical terminology interpretation, and data conversion to a standardized format has helped lay the groundwork for a dataset suitable for NLP model training.

Several challenges did emerge during the project, highlighting opportunities for future work and improvements. The sorting and classification of pathology reports by cancer type, while effective in categorizing most reports, highlighted the limitations of using a keyword-based classification algorithm for sorting. The reliance on predefined terms and negation detection highlighted the need for a more adaptive sorting approach, such as implementing an NLP model capable of understanding contextual information within the text to make more informed decisions. Addressing these classification challenges has the potential to improve the reliability and accuracy of the pathology report sorting.

The comparison of three annotation tools used in this project Prodigy, Doccano, and Label-Studio provided a comparative analysis of their usability, annotation efficiency, and setup complexity to determine the most suitable tool for future work on this project. While all three tools proved suitable for annotating cancer pathology reports, differences between user interfaces, and output standardization influenced their overall suitability for medical text annotation. For future work, Prodigy remains the fastest and most effective annotation tool due to the speed at which items can be annotated, provided the user is comfortable with bash commands and can gain access to a Prodigy license.

The manual annotation and standardization process highlighted the importance of structured guidelines for maintaining consistency across annotated reports. While the manual annotation efforts ensured high-quality data, the time-consuming nature of the process presents a clear opportunity for future research to explore semi-automated or active learning approaches. Integrating pre-trained NLP models to assist with annotation could enhance efficiency while still relying on human oversight for quality control.

Overall, this research provides a foundational guideline and dataset for annotating breast cancer pathology report and can contribute to the broader field of medical NLP research. The insights gained from this project serve as a foundation for training an NLP model capable of summarizing key information from these reports, ultimately supporting medical professionals in providing patient care. Future work will focus on building more efficient and reliable semi-automated annotation tools, followed by training the NLP model to more accurately analyze and interpret data from cancer pathology reports.

References

AWS Marketplace. (n.d.). Retrieved November 9, 2024, from https://aws.amazon.com/marketplace/pp/prodview-kpac4xtqkxuqu

Azure Marketplace. (n.d.). Retrieved November 9, 2024, from https://azuremarketplace.microsoft.com/en-us/marketplace/apps/johnsnowlabsinc1646051154808.gen_ai_lab?tab=overview

Chen, J. S., Lin, W.-C., Yang, S., Chiang, M. F., & Hribar, M. R. (2022). Development of an Open-Source Annotated Glaucoma Medication Dataset from Clinical Notes in the Electronic Health Record. *Translational Vision Science & Technology*, *11*(11), 20. https://doi.org/10.1167/tvst.11.11.20

Chen, W.-T., & Styler, W. (2013). Anafora. *Proceedings of the Conference. Association for Computational Linguistics. North American Chapter. Meeting*, *2013*, 14.

Doccano. (n.d.). Retrieved November 9, 2024, from https://doccano.github.io/doccano/

Features—Brat rapid annotation tool. (n.d.). Retrieved November 8, 2024, from https://brat.nlplab.org/features.html

Fu, S., Wang, L., He, H., Wen, A., Zong, N., Kumari, A., Liu, F., Zhou, S., Zhang, R., Li, C., Wang, Y., St Sauver, J., Liu, H., & Sohn, S. (2024). A taxonomy for advancing systematic error analysis in multi-site electronic health record-based clinical concept extraction. *JOURNAL OF THE AMERICAN MEDICAL INFORMATICS ASSOCIATION*, *31*(7), 1493–1502. https://doi.org/10.1093/jamia/ocae101

García Barragán, A. (2023). *Structuring electronic health records of breast cancer with Natural Language Processing*. https://hdl.handle.net/20.500.14352/105570

Generative AI Lab. (n.d.). *John Snow Labs*. Retrieved November 8, 2024, from https://www.johnsnowlabs.com/generative-ai-lab/

Hao, T., Huang, Z., Liang, L., Weng, H., & Tang, B. (2021). Health Natural Language Processing: Methodology Development and Applications. *JMIR Medical Informatics*, *9*(10), e23898. https://doi.org/10.2196/23898

Kocaman, V., & Talby, D. (2021). *Spark NLP: Natural Language Understanding at Scale* (No. arXiv:2101.10848). arXiv. http://arxiv.org/abs/2101.10848

Macri, C. Z., Teoh, S. C., Bacchi, S., Tan, I., Casson, R., Sun, M. T., Selva, D., & Chan, W. (2023). A case study in applying artificial intelligence-based named entity recognition to develop an automated ophthalmic disease registry. *GRAEFES ARCHIVE FOR CLINICAL AND EXPERIMENTAL OPHTHALMOLOGY*, *261*(11), 3335–3344. https://doi.org/10.1007/s00417-023-06190-2

Montani, I., & Honnibal, M. (2017, August 4). *Prodigy: A new tool for radically efficient machine teaching · Explosion*. https://explosion.ai/blog/explosion.ai

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: An introduction. *Journal of the American Medical Informatics Association: JAMIA*, *18*(5), 544. https://doi.org/10.1136/amiajnl-2011-000464

Natural Language Processing in Healthcare (n.d.). ForeSee Medical. Retrieved November 8, 2024, from https://www.foreseemed.com/natural-language-processing-in-healthcare

Neupane, A. (2023). Literature Survey Paper on Natural Language Processing (NLP) in Voice Assistants. *International Journal for Research in Applied Science and Engineering Technology*, *11*(9), 1411–1419. https://doi.org/10.22214/ijraset.2023.55860

Patil, C. S. (2022). NLP Assisted Text Annotation. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, *06*(06). https://doi.org/10.55041/IJSREM14651

Prodigy. (2024, October 8). https://prodi.gy/docs/prodi.gy

Pyysalo, S., Dunietz, J., & Nuemann, A. (2021). *BRAT (BRAT Rapid Annotation Tool)* [Python]. The Natural Language Processing Laboratory. https://github.com/nlplab/brat (Original work published 2011)

Reiner, B. (2015). Strategies for Medical Data Extraction and Presentation Part 2: Creating a Customizable Context and User-Specific Patient Reference Database. *Journal of Digital Imaging*, *28*(3), 249. https://doi.org/10.1007/s10278-015-9794-4

Santos, T., Tariq, A., Gichoya, J., Trivedi, H., Banjeree, I. (2022). Automatic Classification of Cancer Pathology Reports: A Systematic Review. *Journal of Pathology Informatics.* https://doi.org/10.1016/j.jpi.2022.100003

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., & Tsujii, J. (2012). brat: A Web-based Tool for NLP-Assisted Text Annotation. In F. Segond (Ed.), *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 102–107). Association for Computational Linguistics. https://aclanthology.org/E12-2021

Su, Y., Babore, Y.B. & Kahn, C.E. A Large Language Model to Detect Negated Expressions in Radiology Reports. *Imaging. Inform. med.* (2024). https://doi.org/10.1007/s10278-024-01274-9

Stryker, C & Holdsworth, J. (2024). *What is NLP?* IBM. Retrieved April 10, 2025, from https://www.ibm.com/think/topics/natural-language-processing

Tkachenko, M., Malyuk, M., Holmanyuk, A., & Liubimov, N., 2020- 2022. Label Studio: Data labeling software. Open-source software available from https://github.com/heartexlabs/label-studio

Turner, R. J., Coenen, F., Roelofs, F., Hagoort, K., Harma, A., Grunwald, P. D., Velders, F. P., & Scheepers, F. E. (2022). Information extraction from free text for aiding transdiagnostic psychiatry: Constructing NLP pipelines tailored to clinicians' needs. *BMC PSYCHIATRY*, *22*(1), 407. https://doi.org/10.1186/s12888-022-04058-z