Named Entity Recognition for Case Narratives of Adverse Event Reports

Jayant Yadav
Abstract

In the field of pharmacovigilance (PV), signal detection and assessment activities play a crucial role. They require a PV assessor to read through countless adverse event reports which is manual labor-intensive work. To ease the reading process, visual highlighters can be provided by leveraging natural language processing techniques. These can help in focusing on key information in free-text narratives of adverse event reports. This thesis project is an attempt to address how the existing information extraction tools be fine-tuned to support signal assessment of these adverse event report narratives. To accomplish this, an annotation guideline and gold-standard dataset were created. Information extraction tools namely MedspaCy, CLAMP and Stanza were explored. Named entity recognition models were developed using these tools to extract five entities of interest namely Adverse Event, Drug, Negation, Date and Problem. Subsequently, the models were evaluated on common performance metrics resulting in the highest scoring model with 81.98% F1 score. Additionally, an interactive user interface was also developed for these named entity recognition models and the assessment of its impact on PV assessors was identified as a potential avenue for future work.
Acknowledgements

I would like to give my sincere gratitude to all those who supported me throughout my master thesis journey because of which I was able to complete my work successfully. I would also like to thank my past self for maintaining a positive attitude and unwavering belief in my abilities.

My first token of gratitude goes to my thesis supervisor, Joana Rosado Coelho (PhD.) and co-supervisor, Eva-Lisa Meldau (MSc.) from Uppsala Monitoring Centre (UMC) for being fully supportive and guiding me all throughout. Their expertise, patience and kindness were instrumental in giving me and this thesis the right direction that was needed. Their continued guidance helped me perform at the best of my abilities and take pride in the high-quality work that I have produced.

I would also like to thank my thesis reviewer Professor Matteo Magnani for his invaluable feedback on my work and for making sure that my thesis goes as smoothly as possible. My opponent, Agnes Davidsdottir, for the wonderful and healthy discussion during my thesis defence and for providing me her feedback. Finally, my examiner, Professor Mats Daniels, who conducted my defence and gave his valuable feedback.

I extend my sincere appreciation to my reference group Alem Zekarias, Joana Félix and Qun-Ying Yue for their time and expertise dedicated to this thesis. Their active participation and contribution have been instrumental in shaping the research and the development of the Annotation Guideline. I am grateful for the insightful discussions and brainstorming sessions that helped me understand the pharmacovigilance field. I would also thank the data science team members, including the master thesis student, Erik Norèn, who I shared my office with, for their encouraging and lighthearted talks during countless coffee breaks. Finally, all my colleagues who attended my presentations, asked thought-provoking questions and gave constructive feedback which helped me refine my work.

Lastly, I am indebted to my parents, sister, family members and friends, whose unconditional love, support and belief in my abilities constantly motivated me in my academic journey and continues to do so.

To sum up, I am truly grateful to Uppsala University for imparting me with knowledge, to UMC for providing me resources to test my boundaries and explore myself, and to all those who supported, guided and encouraged me in my journey to true knowledge.

Jayant Yadav
Uppsala, Sweden
June 2023
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLAMP</td>
<td>CLAssification Module for Adverse drug reactions</td>
</tr>
<tr>
<td>MedspaCy</td>
<td>MedSpaCy, a clinical NLP toolkit for biomedical text mining</td>
</tr>
<tr>
<td>Stanza</td>
<td>A general-purpose natural language processing toolkit</td>
</tr>
<tr>
<td>CLAMP</td>
<td>CLAssification Module for Adverse drug reactions</td>
</tr>
<tr>
<td>FastText</td>
<td>A local version of the FastText word embedding model</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>Bidirectional Long Short-Term Memory network</td>
</tr>
<tr>
<td>ICSR</td>
<td>International Classification of Side Effects</td>
</tr>
<tr>
<td>VigiBase</td>
<td>VigiBase is a global database of side effects</td>
</tr>
<tr>
<td>Reservoir Sampling</td>
<td>A technique for estimating the number of distinct elements in a large set</td>
</tr>
</tbody>
</table>

## Contents

1. **Introduction**
   1.1 Objectives
   1.2 Background
   1.2.1 Pharmacovigilance
   1.2.2 VigiBase
   1.2.3 Signal detection and assessment
   1.2.4 ICSR
   1.3 Related Work

2. **Theory**
   2.1 Named Entity Recognition
   2.2 Clinical NLP Tools
   2.2.1 MedspaCy
   2.2.2 Stanza
   2.2.3 CLAMP
   2.3 Word vectors
   2.3.1 Word2Vec
   2.3.2 FastText
   2.4 CNN
   2.5 LSTM
   2.5.1 BiLSTM
   2.5.2 Character-level LM
   2.6 BERT
   2.6.1 Clinical BioBERT
   2.7 RoBERTa
   2.8 Conditional Random Fields
   2.9 Transition Based Parser
   2.10 Reservoir Sampling
   2.10.1 Weighthed Random Sampling - ES
   2.11 Evaluation Metrics
   2.11.1 Precision
   2.11.2 Recall
   2.11.3 F1-score

3. **Data**
   3.1 Composition
   3.2 Dataset Preparation
   3.2.1 Report Exclusion
   3.2.2 Sampling Strategy
   3.2.3 Annotation
   3.3 Dataset Split

4. **Methodology**
   4.1 Selection Criteria
Acronyms

AE  Adverse Event
ADR  Adverse Drug Reaction
BERT  Bidirectional Encoder Representations from Transformers
BiLSTM  Bidirectional Long Short-Term Memory
BioBERT  Biomedical BERT
CLI  Command Line Interface
CNN  Convolutional Neural Network
CRF  Conditional Random Field
EU  European Commission
ICSR  Individual Case Safety Report
LSTM  Long Short-Term Memory
MedDRA  Medical Dictionary for Regulatory Activities
MIMIC  Medical Information Mart for Intensive Care
ML  Machine Learning
NER  Named Entity Recognition
NLP  Natural Language Processing
PIDM  Programme for International Drug Monitoring
POS  Parts Of Speech
PV  Pharmacovigilance
RoBERTa  Robustly Optimized BERT Training Approach
TBP  Transition Based Parser
UI  User Interface
UMC  Uppsala Monitoring Centre
WHO  World Health Organization
WRS  Weighted Random Sampling
1. **Introduction**

With the advancement of technology, the global medicine market has experienced significant growth. Each year, we witness the introduction of new drugs that contribute to this expansion. A study conducted in 2022 [1] revealed that the expenditure on medications in the United States alone amounted to $39.6 billion, marking an 8.4% increase compared to its previous year. The impact of the COVID-19 pandemic has further intensified the efforts of healthcare systems and policymakers worldwide, who remain vigilant in ensuring the safety of these novel medicines and meeting the rising demands in the market.

There have been systems put into place for post-marketing surveillance that ensure patient safety with respect to the use of these medicines and thereby improve the quality of the patient’s life. Pharmacovigilance (PV) plays a pivotal role in this process. Its significance was highlighted in 1961 with the discovery of thalidomide’s link to congenital malformations in babies, as documented by Dr. McBride [2]. Since then, PV has undergone numerous revisions to improve its practices and the definition of adverse events.

As of January 2023, under the WHO Programme for International Drug Monitoring (PIDM), adverse event reports, also referred to as Individual Case Safety Reports (ICSR) (section 1.2.4), are shared by many organizations from all across the globe and stored in VigiBase, the WHO global database of ICSRs [3]. This database is maintained and developed by Uppsala Monitoring Centre (UMC). As the number of medications entering the market, as well as the adverse event reporting continues to rise, the implementation of automation techniques can support PV activities thereby alleviating the clinical and economic burden on healthcare systems globally to some degree [4].

Natural Language Processing (NLP) techniques offer various opportunities to develop and enhance computational methods for signal assessment and detection tasks in pharmacovigilance. One such application of NLP is information extraction which, by performing linguistic analysis and pattern recognition in unstructured free-text of adverse event reports, can help detect entities of interest such as drug names and adverse event mentions. However, in a clinical setting, access to training data is restricted due to data protection regulations aimed at safeguarding patient’s privacy. As a result, very few specialized NLP tools exist which can perform information extraction tasks reliably.

For the PV assessors, interacting with the ICSRs and reading through lengthy clinical narratives is an important step in signal assessment. By leveraging suitable NLP techniques like Named Entity Recognition (NER), relevant entities of interest can be identified and extracted, enabling them to focus on key information with the aid of visual highlighters.

1.1 **Objectives**

This thesis aims to address the research question: "How can the existing information extraction tools be fine-tuned to support signal assessment of adverse event report narratives?". Important aspects of analyzing this question include the identification of the following entities in the free-text field of adverse event reports using NER:

1. Adverse Event (AE)
2. Drug
3. Negation
4. Date
5. Problem
A detailed description with examples of these five entities can be found in Annotation Guideline A.

The following key activities were recognized to achieve the above objective:
1. Prepare a gold-standard dataset
2. Perform exploratory analysis of available IE tools, namely: MedspaCy, CLAMP and Stanza.
3. Develop models using the IE tools
4. Evaluating the IE tool on gold-standard annotated free-text narratives

1.1.1 Delimitations
NER is a well-known application in NLP applied in various domains. However, due to a lack of open-sourced clinical data and gold-standard datasets, unlike biomedical corpora, most of the popular available NLP models are not made to process multilingual clinical texts. Therefore, this thesis focuses only on English narratives and analysis of other languages are for future work.

Developing novel machine learning models to extract entities is also out of the scope of this thesis. Rather, this thesis relies on established NER tools in the medical field, which are developed specifically for clinical texts that are either publicly available or through a license. The reason for using only the clinical NER tools is that when most people try to utilize the state-of-the-art neural architectures pretrained on general domain text data into the clinical domain, it leads to an unsatisfactory performance owing to domain-shift [5]. As evidenced from NLP clinical challenge [6], the most successful clinical NER tools are not just trained on domain-specific treebanks and datasets but also fine-tuned using drug dictionaries and other rule-based string matching algorithms.

Relationship extraction, an application of NLP, is also not in the scope of this work since finding relationships among drugs, AEs and other entities is a challenging topic that would require the involvement of PV experts and is therefore reserved as future work.

1.2 Background
1.2.1 Pharmacovigilance
Pharmacovigilance is defined by the European Medicines Agency (EMA) as "Science and activities relating to the detection, assessment, understanding and prevention of adverse effects or any other medicine-related problem." [7]. Two important activities under PV are signal detection and assessment, where PV assessors try to identify yet unknown adverse events to medicines and try to establish whether there is a causal relationship between the adverse events (AE) and the medicine based on ICSRs.

1.2.2 VigiBase
VigiBase is the largest global database of reported adverse events with over 36 million records collected from more than 170 member countries and its first report dating back to 1968, when WHO PIDM was first established. It is maintained and developed by UMC on behalf of WHO. Within which, drugs get coded to WHO Drug Dictionaries (WHODrug) [8] and adverse events to Medical Dictionary for Regulatory Activities (MedDRA) [3]. MedDRA® terminology is the international medical terminology developed under the auspices of the International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH).

1.2.3 Signal detection and assessment
Signal management is a core activity at UMC, which aims at "identify[ing] and describ[ing] suspected harm to patients caused by the medicines they use" [9–11]. This requires screening through
VigiBase to look for suspected adverse drug reactions that previously went unrecognized or were incompletely documented. A step-wise methodology during signal detection is followed, which can be summarized (not exhaustive) as follows:

1. Leveraging a combination of data mining techniques and clinical evaluation of reports to prioritize medicine-adverse event combinations.
2. In parallel, performing qualitative screening of scientific literature for additional insights into the signal.
3. Individually addressing medicine-adverse event combination for in-depth assessment.

After a signal has been detected, a clinical assessment is performed to assess the likelihood that the suspected adverse event was caused by the medicine.

1.2.4 ICSR

Individual Case Safety Reports (ICSRs), also known as adverse event reports, are stored in VigiBase following certain guidelines [12] that define the data elements for ease of storing and transmission of ICSR among pharmaceutical companies, regulatory authorities and other organizations. These ICSR have a mix of structured and unstructured data - in particular free-text clinical narratives, summaries of patient history, sender and reporter comments. Below is a clinical narrative sample from a published case report of a 13-year-old boy with olanzapine-induced rhabdomyolysis with concomitant lithium-induced electrocardiogram changes:

On day 1 (8 weeks prior to hospitalization) the patient had been admitted to a psychiatric residential care controlled environment facility because of a behavioural disorder and was placed on olanzapine 2.5 mg/d. On day 6, he reported weakness, sore throat, abdominal cramping, myalgias, and diaphoresis, which appeared to be consistent with influenza. On day 14, he experienced increasing weakness and failed to participate in organized activities, which was misinterpreted by the residential care faculty as manifestation of disobedient and oppositional behaviour. Sertraline 100 mg every morning and lithium sustained release 300 mg twice a day were added to olanzapine. On day 27, he fell several times while trying to get out of bed. He was transferred to an outside institution where he was noted to have an elevated creatine phosphokinase, leukocytosis and T-wave inversion of his precordial leads.

According to the guideline [12], these narratives are the source of a "Focused, factual and clear description of the case", which makes them of the utmost importance in the third step of the signal detection activity. The reporter’s comment consists of "diagnosis, causality assessment or other issues considered relevant" and the sender’s comment "describe disagreement with, and/or alternatives to the diagnoses given by the initial reporter". Analysis done on 50 reports containing such narratives revealed that causality or clinical assessment of at least every second report was affected when taking narratives into consideration [14]. More discussion on structured and unstructured fields of interest is done in section 3.

For the definition of AE, please refer to the Annotation Guideline A developed in consensus with the experts at UMC to support this thesis work.

1.3 Related Work

Few of the earliest attempts to develop a text extraction tool at UMC were made in 2009 for extracting ADR information using algorithms based on string matching on permutation of words and spelling variations [15, 16]. Pre-processing techniques such as stop word removal, stemming and synonyms were used. Soundex, Levenshtein distance and Longest common subsequence distance were used for free-text matching. For the medical terminology text search, WHO-ART
and MedDRA dictionaries were employed. The algorithms were further evaluated on free text descriptions of ADRs from Martindale and product labels, achieving 87% recall and 99.96% accuracy. However, these works did not address the extraction of other relevant entities for signal detection like drug names, negation elements and problems. Additionally, the algorithms were not tested on UMC’s own dataset of ICSRs, which includes narratives written by both clinical experts and individuals without specialized knowledge. These individuals may struggle to articulate the adverse events following medical terminology, rendering dictionary lookups or string matching as insufficient approaches.

There have been several other notable investigations conducted at UMC that specifically addressed the extraction of Adverse Drug Reactions (ADRs) from product labels using Bidirectional Long Short-Term Memory (BiLSTM) encoder [17] or by mapping medical verbatim expressions of ADRs to MedDRA terms using a BERT model [18]. Tagging of personal identifiers like person name in case narratives have also been attempted as an effort to remove them from analysis, using deep-learning and rule-based methods [19, 20]. But, extracting the five aforementioned entities from free-text narratives and developing a tool has not been attempted yet according to the author’s best knowledge.

In scientific literature, there are reports of other attempts to develop NER systems that can extract mentions related to adverse drug reactions, negation terms, severity, and drug classes. These systems have been applied to diverse sources such as adverse event reports, biomedical literature, electronic health records (EHR), and social media [21–26]. Notably, teams participating in the 2017 Text Analysis Conference (TAC) challenge achieved significant success in extracting mentions from annotated drug labels, with the highest achieved F1 score of 82.48% [27].
2. Theory

2.1 Named Entity Recognition

Named Entity Recognition (NER) is a Natural Language Processing (NLP) task of identifying named entities in text and classifying them into predefined categories, like Person, Organization and Location. NER is an important task and has various applications in information extraction, question answering, text summarization, sentiment analysis and so on, where NER helps understand the context and extract relevant entities in a large amount of textual data. NER can be extended to more complex tasks such as entity linking, disambiguation and relation extraction for further discovering the relations between named entities and linking them to unique references in the knowledge base.

![Figure 2.1. An illustration of NER task on a news article headline.](image)

A typical NER task consists of three sub-parts: a sequence labeling task, feature space construction, and NER evaluation [28]. Sequence labeling refers to assigning a sequence of tokens to its labels i.e., named entity classes. The system first learns the classification patterns in a set of pre-annotated word labels by extracting distinctive features. It then uses this knowledge to infer labels for unseen sequences of tokens. The input of word labels is usually converted into a more interpretable form using different tagging schemes such as IOB, where each token is marked as either inside (I), outside (O) or beginning (B) of an entity. A few other variants of this scheme are IOB2, BILOU and BIOES [29]. Feature space, for NER task, is usually observed at three levels: words, sentences and documents. At the morphological level, feature space consists of a word, its base form, part-of-speech(POS), prefixes or suffixes and other word patterns. At the contextual level, features represent the presence/absence of trigger words (eg. "Mr." preceding a person). At the document level, features consist of the position of the entity in the document or other document metadata. Finally, the evaluation of NER system is performed by calculating common metrics like precision, recall and F-scores. These scores can be calculated on a partial match, exact match, by giving weights to entities, or a combination of all.

NER methods

The task of NER and its methods have quickly grown over the past 30 years when the concept was first introduced at 6th Message Understanding Conference [30] in 1995. The NER methods can be broadly categorized into three categories, namely, rule-based, machine learning based and deep learning approaches [28, 31]:

1. Rule-based approach: This set of techniques was the first to get implemented over 30 years ago. It consisted of forming use-case specific grammar rules and pattern matching expressions that would overfit a very specific structured text corpus for which it was built. The advantage of this approach was that it was easily interpretable and did not require any labeled dataset to be trained. Tokenization, sentence splitting, syntactic tagging, and trigger word association, were some of the preprocessing steps deployed in rule-based models.
2. Machine Learning based approach: In the late 1990s, this set of techniques started producing state-of-the-art results and dominated over rule-based approaches. Large publicly available annotated text corpora were one of the main reasons that boosted the research in machine learning approaches. Some of the notable techniques under this approach were supervised learning methods like Decision Trees [32], Support Vector Machines (SVM) [33], Hidden Markov Models (HMM) [34], Conditional Random Fields (CRF) [35]. With the need to incorporate more contextual features, unsupervised learning methods were also explored. This included clustering methods like k-means [36], brown-clustering [36] and shallow neural networks in combination with POS tags [37] and skip-gram model (Continuous Bag of Words) [38] as features vectors.

3. Deep Learning based approach: Recently, state-of-the-art results have been reported by models with many hidden layers that are capable of keeping the context information of long sequences rather than just relying on short context windows. The hidden layers can extract features rather than relying on hand-engineered features as in the case of rule-based and machine learning approaches. The most notable architectures used in this approach are convolution neural networks (CNN) [39], recurrent neural networks (RNN), long short term memory (LSTM) [40] either in vanilla or in their bi-directional versions. CRF has also been used as a decoder on top of the previously mentioned networks to solve NER task. Finally, self-attention networks i.e., transformers and its stacked pretrained encoder architectures, which are variants of Bidirectional Encoder Representation from Transformer (BERT) [41] have been highly successful in extracting relevant information from large contexts and solving NER task.

2.2 Clinical NLP Tools

Out of many NLP tools available, the most promising ones in the clinical domain are MedspaCy, Stanza and CLAMP. These tools and their pipelines used in this work are discussed in the following subsections.

2.2.1 MedspaCy

MedspaCy [42] is a clinical Natural Language Processing (NLP) tool provided by the creators of SpaCy, a widely-used NLP toolkit known for its diverse range of pipelines for NLP tasks. MedspaCy is built on the foundation of SpaCy’s architecture, with additional modules and extensions specifically designed to enhance the NLP pipelines for clinical texts.

Framework

The main data structures in MedspaCy are Language, Vocab and Doc objects (Figure 2.2). The Doc objects stores the tokens and the annotations, allowing access through Token and Span objects. An annotation, in the context of NER, refers to the entity names or labels and their corresponding span information. The Example object stores two Doc objects, one for performing predictions on and another as a reference/ground truth. MedspaCy has additional attributes in Doc to the default ones provided in spaCy and can be referred under "spaCy Extention API" block in figure 6.1.

Components

The text processing pipeline (Figure 2.3) comprises several built-in and customized components that are applied sequentially on a Doc object. The first component, Tokenizer, constructs a Doc object by tokenizing the input text. The Tok2vec/transformer component utilizes either a custom
convolutional neural network (CNN) or a transformer-based model (such as Roberta) to generate word embeddings for each token. The NER component, a transition-based parser (TBP), takes input from Tok2vec/transformer component and learns a target representation i.e. class label of non-overlapping spans of tokens. The final predictions are finally stored back in the Doc object.

Figure 2.3. Text processing pipeline utilizing MedspaCy components. ‘...’ stands for any other component that can be attached to the pipeline if needed.

2.2.2 Stanza

Stanza [44] is a python interface to Stanford CoreNLP [45]. It is an NLP toolkit with neural pipelines to perform text analytics, including tokenization, multi-word token (MWT) expansion, lemmatization, part-of-speech (POS) and morphological features tagging, dependency parsing, and named entity recognition.

Framework

Similar to MedspaCy’s data structures, Stanza’s main python objects are Document, Sentence, Token and Word (Figure 2.4). Document object hold the entire annotated document. It holds the Sentence object and can be generated by calling the pipeline. Token object contains the list of syntactic words, their offsets and their NER tags.
Components

Stanza’s text processing pipeline (Figure 2.5) is very similar to MedspaCy’s except that the NER component only works on transformer-based models for generating word embeddings. This component also allows to plug-in it’s own character-level LSTM model (CharLM, section 2.5.2) for generating character-level contextualized word embeddings. The word embeddings from transformer-based model and character-level LSTM are first sent through Bi-directional LSTM (section 2.5.1) after which token classification is performed by Conditional Random Field (CRF, section 2.8) model. It is to be noted that, unlike MedspaCy, transformer and NER components in Stanza are inseparable and effectively considered as the same component.

Framework

In order to be compatible with another NLP system, cTAKES™, CLAMP has been built on the Apache Unstructured Information Management Architecture (UIMA) framework. It also supports asynchronous processing in a distributed environment.
Components

CLAMP’s components were specifically designed to tackle the challenges of NLP in clinical settings. They offer both machine learning-based and rule-based functionalities, including tokenization, sentence boundary detection, POS tagging, section header identification, NER, assertion and negation, among others. The NER component allows users to first select the features (Feature extractor component) they want to extract and use for classification, employing Conditional Random Fields (CRF) with optimization using the L-BFGS approach. This ML component is built using the CRFsuite C++ library [48]. A list of features and an explanation of the L-BFGS method are provided below.

Feature Extractor components

In the NER pipeline, this particular component utilizes multiple techniques to extract features (figure 2.8) from the annotated dataset. These techniques include UMLS dictionary lookup [49], linguistic patterns, word vectors, and clustering algorithms trained on the MIMIC II dataset [50] from the SemEval 2014 Challenge [51]. The linguistic patterns extracted from a report encompass various elements such as POS tags, prefixes, suffixes, sections, sentence patterns, word patterns, and word shapes. The clustering algorithms employed by this component include brown clustering [52] and random indexing [53].

L-BFGS

Limited memory - BFGS (Broyden, Fletcher, Goldfarb, and Shanno) algorithm [54] is an extension of BFGS optimization method. This optimization is used to either minimize or maximize an objective function by approximating its second-order derivative form (Hessian matrix). Since calculating an inverse of Hessian matrix to determine the direction and step size of the optimizer
can be computationally expensive, L-BFGS makes a simplified version of this matrix in order to compute its approximation.

2.3 Word vectors

Word vectors are numerical representations of words that try to capture their meaning in a natural language text. These representations are often learned from word distributions in a self-supervised fashion and are widely used in NLP tasks. The word vectors can either be constructed by models (for instance GloVe) that leverage global co-occurrence statistics of the words appearing in the text or by using prediction-based models (for instance Word2Vec and FastText) that model the probability of a word occurring given a sequence of words. [55]

2.3.1 Word2Vec

Word2Vec [38] is a widely used method that generates static word embeddings for NLP tasks. This work by Google introduced continuous bag-of-words (CBOW) and skip-gram (SG) models that could be trained on very large corpora in a short time giving accurate high-dimensional word vectors that were helpful in semantic and syntactic language tasks. CBOW and SG are both log-linear language models but differ in their objective functions. While CBOW aims to predict the center/target word based on its context, SG performs the reverse task of predicting context words based on a center word. The main advantage of these models are that they show linear semantic inference for example ‘France’ - ‘Paris’ + ’Italy’ = ’Rome’.

2.3.2 FastText

The FastText library [56] by Facebook is an improved variant of Word2Vec skip-gram model which generates word embeddings on n-gram level instead of word level. It overcomes the drawback of Word2Vec on out-of-vocabulary (OOV) words by enriching the word vectors with sub-word information. It thus works better for morphologically rich languages with word composition following some rules for instance Finnish, Spanish and Turkish language.
2.4 CNN

Convolutional Neural Networks (CNN/ConvNet) are widely known for their image-processing capabilities but they can also be applied to text inputs for applications such as text classification or sentiment analysis [57]. The two main operators in CNN are convolution and pooling. The convolution operator matches a pattern in an image, while the pooling operator either locally or globally aggregates the matches over a position. In case of text input, these operators are used as 1D operators. The convolutional layer will apply filters to extract local patterns/features from the text by capturing n-grams at each position. This output is then passed through pooling layers to down-sample the extracted features (Figure 2.9). Finally, the resulting features are fed into fully connected layers for classification or prediction. CNNs model allows capturing local dependencies and patterns in the text, which can be useful for tasks that require an understanding of local contexts, such as NER.

![Figure 2.9. CNN being used for extracting character features from each word for an NER task [58].](image)

2.5 LSTM

Long Short-Term Memory (LSTM) [40, 59] are a special type of Recurrent Neural Network (RNN) [60] which were designed to eliminate long-term dependency problems. That means LSTM can retain information for a longer period of time than RNNs, enabling them to capture dependencies between words or characters over long distances in the input sequence. LSTM is able to achieve this by having a memory unit that contains the cell state and gated units that control the flow of information. These gates [61] can add or remove information to the cell state by the use of sigmoid functions and linear point-wise operations (Figure 2.10).

LSTMs have proven to be effective in learning and capturing patterns in the input text and learning dependencies, making them popular for various NLP tasks like NER. They have been adapted and modified to create multiple versions that can capture patterns at different levels, whether it be at the word or character level, depending on the specific requirements of the task at hand.
2.5.1 BiLSTM

Bidirectional LSTM (BiLSTM/BLSTM) [62] is an LSTM variant made to overcome the limitation of LSTM and RNN wherein their outputs are mostly based on the context appearing previously in the sequence. Since information flows mostly in one direction for a standard LSTM, in BiLSTM, the input sequence is processed in both directions using a forward LSTM layer and a backward LSTM layer. The final representation of the sequence can then be derived by concatenating the outputs from the forward and the backward LSTM layer (Figure 2.11). By performing passes from both directions, the model gets a better understanding of the context from the input sequence. This is particularly useful for NLP tasks such as sequence tagging [63], sentiment analysis and machine translation.
2.5.2 Character-level LM

Character-level Language Model (CharLM) [64] is an LSTM variant that has a forward and backward LSTM layer that reads text as a sequence of characters. Each LSTM cell in the forward and backward layer is then trained to predict the next character using its hidden state. The final embedding of a word is constructed by concatenating information from the first hidden state of the backward LSTM layer and the last hidden state of the forward LSTM layer (Figure 2.12). This model is particularly used for generating contextualized string embeddings on a character level (also known as flair embeddings) to provide input to a sequence labeling task such as NER.

![Figure 2.12. Contextualized string embedding of the token “Washington” using CharLM [64]](image)

2.6 BERT

Bidirectional Encoder Representations from Transformers (BERT) [41] is a language model that is derived from a transformer [65] model by stacking multiple encoder blocks in an attempt to get a deeper understanding of the language context and solve a wide range of tasks. By training BERT using masked language modeling (MLM) and next sentence prediction (NSP), it can perform tasks such as question answering, sentiment analysis, text prediction, NER and text summarization to name a few. Through MLM, BERT learns by masking/hiding words and predicting the masked word based on the context provided by surrounding words. NSP helps BERT understand the relationship between two sentences by predicting if they appear consecutively or not. The training process consists of a pre-training phase, where BERT is trained using MLM and NSP, and a fine-tuning phase, where BERT is further trained with an appropriate output layer for specific downstream tasks such as question answering.

![Figure 2.13. BERT architecture. The pre-training and fine-tuning procedures follow the same architecture except for an additional output layer during fine-tuning [41].](image)
2.6.1 Clinical BioBERT

Clinical BioBERT [66] is a BERT variant whose weights were initialized from BioBERT [67] model. The BioBERT was itself initialized from BERT-BASE, trained on general domain corpora (English Wikipedia [68] and BooksCorpus [69]), and further pre-trained on biomedical domain corpora (PubMed abstracts and PubMed Central full-text articles [70]). The clinical BioBERT was then pre-trained on all notes from the MIMIC III v1.4 dataset. The MIMIC III (Medical Information Mart for Intensive Care) [71] is a publicly available database collected from 2001-2012 and consists of de-identified health-related records of patients submitted to critical care units at Beth Israel Deaconess Medical Center.

The model has 12 transformer blocks, 768 embedding dimensions, 12 attention heads and a total of 110M parameters. It was pre-trained on case-sensitive text of 128 sequence lengths in a batch size of 32 for a total of 150,000 steps. The model is hosted on HuggingFace (a platform with ready-to-use ML model repository) and can be downloaded and integrated into NLP tools like MedspaCy and Stanza.

2.7 RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa) [72] is a recipe for training BERT models that has demonstrated improved performance on various NLP benchmarks and has become a popular choice for downstream tasks like NER. RoBERTa follows a similar pre-training and fine-tuning approach as BERT but incorporates additional modifications during pre-training. These modifications include training the model longer, with bigger batches, with more data and on longer sequences. It also changes the masking pattern to a dynamic one and removes the next sentence prediction (NSP) objective.

The RoBERTa-Base model has the same architecture as Clinical BioBERT since they are both derived from BERT-Base. However, RoBERTa is trained from five different general domain datasets, which are derived from BooksCorpus, English Wikipedia, CC-News, OpenWebText and Stories [68, 69, 73–75].

2.8 Conditional Random Fields

Conditional Random Fields (CRF) [76] are probabilistic models for structured prediction. In other words, they combine classification and graphical modeling to predict variables (for example labels) that have dependencies on each other and with the observed variables (for example tokens/words). CRFs have been found to be very useful in sequence tagging tasks in NLP [35]. The dependence (transition probabilities) of labels might not only come from the current token but also from previous or future tokens in the sequence. Further, the features of these tokens can be arbitrary, overlapping or agglomerative in nature (for instance features from words, characters or features from text layout). CRFs are a popular choice of discriminative models for predicting sequence labels.

2.9 Transition Based Parser

Transition Based Parsers (such as arc-standard and eager-arc parsers) [77] are algorithms that perform syntactic parsing over natural language text. They do so by constructing a dependency tree for the given text by incrementally applying transition rules or actions in one left-to-right sweep over it. The parser maintains two stacks (designated as "output" and "stack") and a buffer containing tokens/words yet to be processed. While stepping through each token in the text, the parser asks a "guide" or an "oracle" which transition actions to apply until the algorithm reaches a termination state. Typically, this oracle is a ML algorithm that takes the state of the parser, calculates action probabilities and outputs an action in order to generate a new state.
One of the possible actions for which the oracle outputs the action probability is the $SHIFT(y)$ transition. This action pops tokens from the top of the stack, labels them with label $y$ and pushes them into the "output" stack. For a NER task, these labels can represent a class to be given to a chunk of tokens, for instance, \textit{B-PERSON} or \textit{I-LOC}.

\textbf{2.10 Reservoir Sampling}

Reservoir sampling is a set of algorithms for randomized selection without replacement. The total population size is either very large, unknown or a data stream and a fixed number of elements are selected out of them. These algorithms are particularly useful when the total dataset cannot fit into the memory size.
2.10.1 Weighted Random Sampling - ES

Weighted random sampling algorithm developed by Efraimidis and Spirakis (WRS-ES algorithm [78]), is one of the WRS approach that comes from the family of reservoir sampling algorithms. The idea behind this type of sampling is to select $m$ weighted random samples without replacement out of a population $V$ with $n$ weighted items ($n >> m$) where their relative weight determines the probability of getting selected. If the weight of item $v_i$ is $w_i$ and $\text{random}(L,H)$ is a function that generates a uniform random number in $(L,H)$ then the algorithm can be given as follows:

<table>
<thead>
<tr>
<th>Algorithm 1: WRS-ES algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data:</strong> A population $V$ of $n$ weighted items</td>
</tr>
<tr>
<td><strong>Result:</strong> A WRS of size $m$</td>
</tr>
<tr>
<td>for $v_i \in V$ do</td>
</tr>
<tr>
<td>$u_i = \text{random}(0,1)$</td>
</tr>
<tr>
<td>$k_i = u_i^{1/w_i}$</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>Select the $m$ items with the largest keys $k_i$ as a WRS</td>
</tr>
</tbody>
</table>

2.11 Evaluation Metrics

In classification tasks, accuracy, precision, recall, and F-score are commonly used evaluation metrics. To visually assess the performance of a model, a confusion matrix is generated. The confusion matrix organizes the data, with each row representing an actual class and each column representing a predicted class and vice-versa. For a binary-classification task, the matrix looks like the table 2.1:

<table>
<thead>
<tr>
<th>Predicted \ Actual</th>
<th>Positive label</th>
<th>Negative label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive label</td>
<td>True Positive (TP)</td>
<td>False Negative (FN)</td>
</tr>
<tr>
<td>Negative label</td>
<td>False Positive (FP)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

Table 2.1. Confusion matrix for a binary classification

In multi-classification tasks, relying solely on accuracy as a performance measure can be misleading, particularly when there is class imbalance. Accuracy tells us how many predictions were correct out of the total predictions made by the model. If a model excels in predicting the majority class but struggles with minority classes, the accuracy metric may appear high, giving a false impression of overall performance.

2.11.1 Precision

Precision, also known as Positive Predictive Value (PPV), is a metric that indicates the proportion of true positive predictions out of all positive predictions made. In binary classification, precision can be calculated using the formula 2.1, where $TP$ represents the count of true positives and $FP$ represents the count of false positives from the given table 2.1.

$$
\text{Precision} = \frac{TP}{TP + FP} \tag{2.1}
$$
In a multiclassification task with \( n \) classes, precision can be calculated by averaging the precision values across all classes. When precision is calculated individually for each class using True Positives and False Positives, it is known as micro-averaging (eq 2.2). On the other hand, if precision is calculated for each class separately and then averaged, it is referred to as macro-averaging (eq 2.3).

\[
\text{Micro Precision} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \tag{2.2}
\]

\[
\text{Macro Precision} = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i} \tag{2.3}
\]

### 2.11.2 Recall

Recall, also known as hit rate or sensitivity, is a metric that indicates the proportion of true positive predictions out of all positive cases from the ground truth. Like precision in binary classification, recall can be calculated using the equation (2.4) where \( FN \) represents False Negatives.

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{2.4}
\]

Similar to precision in multiclassification task with \( n \) classes, recall can be calculated as a micro-average (2.5) or macro-average (2.6) value.

\[
\text{Micro Recall} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \tag{2.5}
\]

\[
\text{Macro Recall} = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i} \tag{2.6}
\]

### 2.11.3 F1-score

The F-score metric combines precision and recall to give a model’s accuracy. The most common form of F-score is the F1 (or \( F_1 \)) score which is the harmonic mean (eq 2.7) of precision and recall and gives them equal weightage.

\[
F1-score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.7}
\]

In case of a multiclassification task with \( n \) classes, F1-score takes the similar form of micro-average (eq 2.8) and macro-average (eq 2.9) values like precision and recall.

\[
\text{Micro F1-score} = \frac{\text{Net}TP_i}{\text{Net}TP_i + \frac{1}{2}(\text{Net}FP_i + \text{Net}FN_i)} \tag{2.8}
\]

\[
\text{Macro F1-score} = \frac{1}{n} \sum_{i=1}^{n} \text{F1-score}_i \tag{2.9}
\]

where \( \text{Net}TP, \text{Net}FP, \text{Net}FN \) represent the sum of \( TP, FP, FN \) respectively for each class. To calculate them, the multi-class confusion matrix is first "flattened" in the axis of the class for which \( TP/FP/FN \) are being calculated.
3. Data

This section illustrates how the final annotated dataset of 450 records was prepared from a frozen version of VigiBase containing over 36M unique records up to date January 2023. A reference group was setup consisting of three domain experts. Their involvement was instrumental in understanding the PV science field, VigiBase and for drafting the annotation guideline.

3.1 Composition

To prepare training data for the NER task of identifying the five entities in question, selecting a diverse set of reports representative of the entire VigiBase was essential. Further, these reports were also supposed to contain the most informative narratives so that the NER models could learn more from the high-quality narratives. Understanding which reports would contain more quality narratives was done under the guidance of the reference group and supervisors who were well-versed in the characteristics of such reports.

To ensure the dataset diversity, it was also essential to take into consideration the large reporting rates related to the COVID-19 vaccines during the pandemic which made up the majority of the reporting during that time period. Therefore, making sure to have a representation of all types of drugs within the dataset was important to ensure that the model would generalize better.

Keeping in mind the objectives of maintaining the diversity of reports and informativeness in the narratives, the following fields were either derived or imported from the full dataset:

1. ID: identifier of a unique ICSR report.
2. Country: country code in which the adverse event was reported.
3. Reporter qualification: qualification status of the reporter, like physician, pharmacist, non-health professional and so on. A report can have multiple reporters, giving rise to a one-to-many relationship.
5. Company report flag: indicates whether the report was identified as having been submitted by a company to the national PV center.
6. Serious flag: indicates whether the report described a serious or non-serious case.
7. Vaccine report flag: indicates whether the report contained at least one vaccine.
9. Clinical narrative text: case narrative which may include clinical course, therapeutic measures, outcome, and additional relevant information.
10. Patient medical history text: relevant medical history such as disease or surgical procedure, including onset and resolution date as well as relevant comments.
11. Sender’s comment text: sender’s assessment of the case.
12. Reporter’s comment text: reporter’s comments on the diagnosis, causality assessment, or other relevant issues.

3.2 Dataset Preparation

Dataset preparation was a multi-step process involving report exclusion, weighted random sampling and finally another step of random sampling for annotating narratives (figure 3.1). Henceforth, the fields 2 - 8 from the previous section, will be collectively referred to as features and individually as feature or feature type. The instances of each feature type, such as ‘USA’ is an
instance of Country feature, will be referred to as either feature instances or instances. Further, a combination of instances, for example ('USA', 'Spontaneous', 'Serious'), will be referred as instance set. For the number formatting, a period (.) is used to separate decimal fractions and a comma (,) is used to separate every three digits.

3.2.1 Report Exclusion

The first and most obvious filtration step was to exclude the reports with duplicates and non-English narratives since our scope is limited to NER of English narratives only.

Duplicate cases and foreign reports were filtered out using special flags stored in VigiBase. Foreign reports refer to the reports where reporting country is different from the country where the AE occurred.

Non-English narratives were filtered out using the language detection python library fastText [79, 80], a text classification model using pre-trained word vectors trained on Wikipedia, Tatoeba and SETimes corpora. The model predicts the top k languages out of 176 languages, where k can be specified by the user. Since we were only interested in detecting the English language, clinical narratives (≥100 characters) with 95% probability of the English language were kept, and the rest were discarded. English language detection was also performed for other narratives, i.e., sender comments, reporter comments, and patient medical history, but was not considered in the final decision of filtering out the reports. This was decided after observing that almost all of these narratives were detected to be in English if the clinical narrative was also in English. After dropping duplicate cases, foreign cases and cases with narrative length <100 characters, roughly 7.7 million records remained. Further, after removing reports with non-English clinical narratives, approximately 1.38 million reports were left to which subsequent sampling steps could be performed.

Figure 3.1. An overview of different stages of the dataset preparation phase and count of reports at each step

The length of the narrative set at 100 characters and threshold of 95% were carefully selected after analyzing a few sample outputs which indicated that the accuracy of fastText (refer section 2.3.2) model decreases as the amount of biomedical and clinical terms in the narrative increases. This was not surprising since the model was never trained on corpus rich in such terms. Excluding shorter narratives helped exclude cases with non-informative narratives, mainly consisting of text such as ‘NA’, ‘Not available’, ‘nothing significant’. Preserving longer text (≥100 characters) for training meant keeping the text sequences long enough for the NER models to derive context behind the clinical terms. Furthermore, excluding shorter narratives from the evaluation was acceptable since the value of highlighting entities applies mostly to texts which are too long to read in a short amount of time.
Other models that were tested for language detection were Naive Bayes (langdetect python library [81]) and a character level Bi-LSTM trained off of text snippets from the Universal Dependencies 2.5 dataset. (MultilingualPipeline module of Stanza python library [44]). There were multiple points of failure for langdetect library. It produces a very low/unsatisfactory probability of a language if the text is short or has a mixture of languages. For instance, when a Portuguese narrative had an English translation appended, the model gave the probability of English very high (> 90%). This may result from how the posterior probability of a text belonging to a certain language is calculated. Based on the assumption that each language has its unique features, i.e., the presence of peculiar characters and spelling rules. The posterior of a language is updated as it encounters the features of its language. The guessed language is the one with the maximum probability. Since these features are not well defined and lack syntactic knowledge of the language, they tend to confuse if the text has a mix of languages. The use of fastText over langdetect brought down the count of English clinical narratives by 4 times (figure 3.2).

![Figure 3.2. Count of reports after English language detection using LangDetect and FastText package](image)

Stanza MultilingualPipeline module, on the other hand, did not meet our expectations because it returns only the language class and not its probability. This restricted the flexibility to play around with the degree of uncertainty we would allow for a language detection task in a narrative.

3.2.2 Sampling Strategy
Performing annotations is a resource-intensive endeavor. As a result, it was imperative to approach the sample selection process meticulously when working with our extensive dataset of 1.38 million records. We aimed to narrow the dataset to a smaller, manageable subset that could be annotated within a reasonable time frame.

This subset had to be kept diverse while maintaining each feature’s diversity in specific proportions. Going back to the Covid report overpopulation problem, the reference group wanted to represent drug reports more than non-Covid vaccine reports followed by vaccine reports. A ratio of 90% : 9% : 1% was decided for Drug : Non-Covid Vaccine : Covid Vaccine reports, which was based on the ratio followed by drug vs vaccine reports pre-pandemic. Similarly, reporting of a serious case is supposed to have a better quality of narrative, hence more informative, than a non-serious case. As a result, serious : non-serious cases were set to be in the ratio of 70% : 30% in the final dataset. The rest of the features (country, reporter qualification, report type, and company report) were set to be in equal ratios.

Informative samples could also be captured by taking the presence of other free-texts, i.e., patient history, the sender’s and reporter’s comments, into consideration. However, coming up with a good ratio for these features was difficult because it would have favored countries that include such free-texts more than the others owing to the different reporting policies they follow.
Given the constraints outlined above, it was evident that random sampling was inadequate for our purposes. Random sampling operates by generating a random number from a uniform distribution, thus providing equal probability for each sample. However, this method would not have produced the desired outcomes given our specific criteria. There was a need to develop a strategy to sample by giving importance to some samples over others based on their features. To tackle this problem, a weighted random sampling (WRS-ES algorithm, section 2.10.1) approach was adopted.

To assign weights \( w_i \) to each sample report \( i \), the weights of each feature instance \( f \) in the feature set \( F \) were initially calculated. If \( p_f \) is the percentage of reports with feature \( f \), \( p'_f \) is the desired percentage of the population with feature \( f \) that we want in our selection (reservoir). \( n_f \) is the count of samples with feature \( f \) and \( n_{total} = n \). Then the weight of feature \( w_f \) and report \( w_i \) can be derived as:

\[
\begin{align*}
   p_f &= \frac{n_f}{n_{total}} \times 100\% \\
   w_f &= \frac{p'_f}{p_f} \quad (3.1) \\
   w_i &= \sum_{f \in F} w_f \quad (3.2)
\end{align*}
\]

Further, key \( k_i \) based on WRS-ES algorithm can be calculated as:

\[
\begin{align*}
   u_i &= \text{random}(0,1) \\
   k_i &= u_i^{1/w_i} \\
   \text{where,} \\
   \text{random}(a,b) &= U(a,b)
\end{align*}
\]

Samples with top 20 keys \( k_i \) were then selected from 1175 instance sets of ('Country', 'Report type', 'Company report flag', 'Reporter qualification'). The count of 20 was empirically chosen such that final sample set has at least one sample from each feature instance while keeping the total count low. An instance set, for example, looked like ('USA',2,1,5) which represented a group of reports belonging to ('USA', 'Report from Study', 'company report', 'Consumer/Non-Health Professional'). Since many instance sets had less than 20 samples, the resultant selection contained 13,262 samples instead of 1175 \( \times 20 \) = 23,500. In these 13,262 selected samples, no conflicting keys \( k_i \) were observed, but contained duplicate reports as a result of one-to-many relationship with 'Reporter qualification'. Hence, the final choice after weighted random sampling was 13,262 samples with 12,884 unique reports, as opposed to 1.38 million reports in the previous Report Exclusion step.

There were two design choices made while selecting samples with higher weights and at the same time making sure they were diverse in features. Firstly, in the original WRS-ES algorithm, \( \text{random}(L,H) \) function generates uniform random numbers in the range \( (L,H) \). However, in equation 3.3, \( \text{random}(L,H) \) function was replaced to generate a random number from a power distribution [82] with probability density function: \( P(x,a) = ax^{a-1}, 0 \leq x \leq 1, a = 5 \). The motivation behind this choice was to select samples with higher weights more than the ones with lower weights in their subgroups, given that the distribution of sample weights was observed to have a lighter tail (figure 3.3). Then secondly, to ensure diversity in selection, samples were grouped under instance set ('Country', 'Report type', 'Company report flag', 'Reporter qualification') and then samples were extracted from each instance set. This prevented WRS-ES from selecting only high-weighted samples which were a result of underrepresented feature instances. For example, countries with smaller datasets like Zambia (~5k reports) might have received extremely high weights because it was underrepresented in the pool of 1.38 million records. To suppress selecting all the reports from such countries, a cap of 20 was put on a combination of the previously mentioned instance set.

This choice of sampling strategy worked for some of the feature instances but did not attain the ideal distribution of ratios that we initially intended to achieve (figure 3.4). A possible reason
is that many feature instances were greatly underrepresented in the initial dataset of 1.38 million. A potential downside of this sampling strategy lies in how subgroups were created. Since a sampling size of 20 was fixed for each subgroup, it might have led to misrepresentation based on the country feature knowing that some countries are significant contributors in VigiBase but could only be represented by a small number of reports in our final dataset. An upside of this strategy, however, is that the NER models will get to learn from all styles of reporting from different countries and hence generalize better. For future considerations, the purpose or the use case of dataset creation should be kept in mind while developing a sampling strategy. Statistical models might benefit from datasets that are more representative of their parent datasets as they learn more from syntactic analysis, and different countries have their way of reporting. Whereas sequential models might benefit more from datasets that have more diverse samples, therefore understand underlying semantics and as a result generalize better.

**Figure 3.3.** Count of reports distributed by weight and belonging to subset: (‘USA’, ‘Spontaneous’, ‘Non-company report’, ‘Unknown reporter qualification’)

**Figure 3.4.** Weights of 20 sampled reports and belonging to subset (‘USA’, ‘Spontaneous’, ‘Non-company report’, ‘Unknown reporter qualification’). Left: Using random sampling. Right: Using weighted random sampling

### 3.2.3 Annotation

To detect the entities in free-text narratives, machine learning algorithms were trained on labeled/annotated dataset. Therefore, the annotation phase consisted of drafting a guideline (See appendix A) followed by annotation of free-text narratives. This phase was performed in parallel with the report exclusion phase. Discussions with a reference group consisting of three adjudicators lead to the basic definitions of each entity to be annotated. The adjudicators performed sample annotations to reach a consensus on these definitions. After that, the annotation guideline went through multiple iterations to reach its final version referring to which annotation was performed by the author alone. An overview of this activity is visualized in figure 3.5. In total, 5 entities were annotated in free-text narratives of 450 ICSR reports. For assistance two annotation tools,
Doccano and Prodigy were compared for their range of functionalities and flexibility so that the final annotated dataset could easily be transformed into inputs for the three NLP tools which were to be evaluated. Prodigy was selected as the annotation tool for its detailed documentation and available packages that allowed to build of highly custom NER datasets. It is also developed by the makers of MedspaCy which meant easier dataset ingestion to built-in models in MedspaCy.

![Guideline development diagram]

**Figure 3.5. Guideline development**

For the ease of annotation and to help improve the contextual awareness of the models, section headers: "SECTION:Summary_NarrativeIncludeClinical", "SECTION:Summary_PatientMedicalHistoryText", "SECTION:Summary_Sen-derComment" and "SECTION:Summary_ReporterComment" were added to the narrative bodies. No assistance from the structured fields of the ICSR was taken by the author to annotate the narratives. For annotation, 450 reports were then randomly selected which preserved the underlying feature distribution (Appendix figure 6.2) achieved during the sampling phase of 12,884 reports. In total, 444 reports had entities and 6 reports had no entity to be annotated. 10,420 entity spans were captured for the 5 said entity types. The distribution of reports per entity and the count of entities per type can be observed in Figure 3.6 and Table 3.4 respectively.

![Distribution of Reports chart]

**Figure 3.6. Count of reports per entity type in an annotated dataset of 450 reports**

During annotation (Appendix figure 6.3), even though the annotation guideline was followed strictly, there were instances when the report samples presented tricky narratives or entities which
were not adequately addressed in the annotation guideline. Since the guideline could not be updated during the annotation process, such cases were handled loosely. A few examples are as follows:

- Discontinuous tokens were not annotated. In "...acute, subacute headache ..." annotated only "subacute headache" span as AE.
- "swollen tongue and mouth" was considered one AE term. However, if the list of swollen areas was longer, only "swollen tongue" was annotated.
- In cases of negation and AE entity conflict, AE span was given precedence. "Health care professional (HCP) confirmed that the drug was not working", "not working" was marked AE and was given precedence over "not" negation.
- "Hb was 60 g/L" was not annotated as AE, even though it is below the normal range. The author was not trained enough to understand what constitutes a normal range in lab reports.

To have a more informative annotated dataset, active learning techniques for report selection were considered but later dropped. Since multiple models were to be trained under the three NLP tools and annotating reports based on active learning would have led the dataset to get biased towards the model on which it was being evaluated for reducing uncertainty.

At the end of the dataset preparation process, an evaluation was performed to make sure no report went unaccounted and to derive the statistics of the final annotated dataset. A summary of the same on dataset and feature level is reported in Table 3.1 and 3.2, respectively.

<table>
<thead>
<tr>
<th>Dataset stage</th>
<th>Total Feature instances</th>
<th>Unique reports</th>
<th>Instance sets</th>
<th>Reports per instance set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full VigiBase</td>
<td>272</td>
<td>36,438,671</td>
<td>-</td>
<td>Min</td>
</tr>
<tr>
<td>Post Report exclusion:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>foreign, dedup. reports</td>
<td>177</td>
<td>7,724,812</td>
<td>7,495</td>
<td>1</td>
</tr>
<tr>
<td>narratives &gt;=100 chars</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post Report exclusion:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English narratives only</td>
<td>142</td>
<td>1,386,608</td>
<td>2,999</td>
<td>1</td>
</tr>
<tr>
<td>Post sampling</td>
<td>142</td>
<td>12,884</td>
<td>2,460</td>
<td>1</td>
</tr>
<tr>
<td>Post Annotation</td>
<td>103</td>
<td>450</td>
<td>451</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.1. Dataset statistics at different stages

<table>
<thead>
<tr>
<th>Features / Dataset stage</th>
<th>Country</th>
<th>Vaccine Report flag</th>
<th>Covid vaccine report flag</th>
<th>Reporter Qualification</th>
<th>Report Type</th>
<th>Serious flag</th>
<th>Company Report flag</th>
<th>Total Feature Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full VigiBase</td>
<td>252</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>272</td>
</tr>
<tr>
<td>Post Report exclusion:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>foreign, dedup. reports</td>
<td>157</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>177</td>
</tr>
<tr>
<td>narratives &gt;=100 chars</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post Report exclusion:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English narratives only</td>
<td>122</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>142</td>
</tr>
<tr>
<td>Post sampling</td>
<td>122</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>142</td>
</tr>
<tr>
<td>Post Annotation</td>
<td>83</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>103</td>
</tr>
</tbody>
</table>

Table 3.2. Feature Instance statistics at different stages
3.3 Dataset Split

In supervised machine learning, the training of models is conducted by splitting the dataset into training, development and test disjoint sets. The development set is used to perform hyperparameter tuning of the models. It is also essential to keep the test set aside before model development so that the models and the person developing them remain unbiased during evaluation. Following suit, 100 reports from the 450 annotated reports were saved as test dataset. To maintain randomness, the test set was selected from the last 100 from the main dataset, which had already been randomly sampled.

Each report in the annotated dataset contains multiple entities, giving rise to a multi-label set. A classification task, such as this, requires stratified splitting over random splitting so that the number of labels of each type falls equally in both train and development split. This is important for avoiding the absence of rare labels in our test sets, especially in an information retrieval task of clinical terms where it is highly likely that the model will see rare terms in production use-cases. Cross-validation is one workaround when dealing with small labeled datasets, but it has been shown that stratification performs better in terms of bias and variance [83]. For the training and development set, stratification was thus preferred over random splitting and cross-validation. In particular, iterative stratification [83] was performed which considers each entity independently to transfer a report into either a training or development set.

The training and development split was set at 75%:25% and to validate that the iterative stratification was indeed giving a better distribution of entities over a random split, the count of each entity type in every split was summarized for 350 annotated reports. It can be observed from the Table 3.3 that for each entity type, its ratio in training and development is much closer to our desired 75%:25% split than what we observe for a random splitting strategy. A final rerun of the dataset split was performed, giving us a distribution of annotated entities summarized in Table 3.4.
### Table 3.3. Split percentage of Train and Dev set with iterative stratification and random sampling

<table>
<thead>
<tr>
<th>Entity</th>
<th>Stratified</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train set</td>
<td>Dev set</td>
</tr>
<tr>
<td>AE</td>
<td>76.56%</td>
<td>23.44%</td>
</tr>
<tr>
<td>DRUG</td>
<td>76.06%</td>
<td>23.94%</td>
</tr>
<tr>
<td>NEGATION</td>
<td>75.46%</td>
<td>24.54%</td>
</tr>
<tr>
<td>DATE</td>
<td>74.80%</td>
<td>25.20%</td>
</tr>
<tr>
<td>PROBLEM</td>
<td>74.28%</td>
<td>25.72%</td>
</tr>
</tbody>
</table>

### Table 3.4. Count of entities per dataset split using iterative stratification on Train+Dev set

<table>
<thead>
<tr>
<th>Entity</th>
<th>Train set</th>
<th>Dev set</th>
<th>Test set</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>1774</td>
<td>543</td>
<td>703</td>
<td>3020</td>
</tr>
<tr>
<td>DRUG</td>
<td>1601</td>
<td>504</td>
<td>555</td>
<td>2660</td>
</tr>
<tr>
<td>NEGATION</td>
<td>1322</td>
<td>430</td>
<td>520</td>
<td>2272</td>
</tr>
<tr>
<td>DATE</td>
<td>1006</td>
<td>339</td>
<td>392</td>
<td>1737</td>
</tr>
<tr>
<td>PROBLEM</td>
<td>439</td>
<td>152</td>
<td>140</td>
<td>731</td>
</tr>
</tbody>
</table>

*Table 3.3. Split percentage of Train and Dev set with iterative stratification and random sampling*

*Table 3.4. Count of entities per dataset split using iterative stratification on Train+Dev set*
4. Methodology

4.1 Selection Criteria

Each of the three tools offers a range of NLP components and word vectors, providing the flexibility to either construct a customized model or utilize a pre-trained NER model with predefined entity types. The selection of components was guided by the recommendations provided by these tools for NER tasks. To assess the model’s extensibility, a third-party transformer model from HuggingFace (a platform with a ready-to-use ML model repository) [84] was incorporated whenever feasible, allowing for further exploration of its capabilities. All the models were built and trained on 12\textsuperscript{th} Intel Core i7-12700H 2.3 GHz CPU.

MedspaCy and CLAMP also give freedom to add custom rules into the NLP pipeline, such as concept recognizers, dictionary lookups for drugs and AEs, optimized tokenizers and sentence detectors for clinical texts. However, with the goal of developing an information extraction tool prototype and conducting a comparative evaluation, these customizations which could have potentially enhanced the models, were not implemented. Moreover, assessing the impact of these rules would have posed considerable difficulties and incurred significant time. This is primarily due to the diversity of sources of the ICSR reports that originate from various hospitals, organizations and counties, submitted by both healthcare professionals and non-professionals. There might be inconsistencies in the reporting styles and adherence to standard conventions that might have made it difficult to assess the effectiveness of the rules and their potential impact on the model’s performance.

For MedspaCy and Stanza, two pretrained state-of-the-art transformer based models, namely RoBERTa and Clinical BioBERT [66], were obtained from HuggingFace resources and utilized. The RoBERTa model exists in two settings: RoBERTa-BASE and RoBERTa-LARGE. To perform a head-to-head comparison with Clinical BioBERT which is initialized from BioBERT using BERT-BASE architecture, the RoBERTa-BASE version was used. This version is also built on the same BERT-BASE architecture consisting of 12 attention heads and 125 million parameters.

4.2 MedspaCy

Building a custom NER model in MedspaCy required input preparation and model training. The input was prepared and stored in a special \texttt{Doc} object, a container for storing and accessing linguistic annotations like tokens, spans, sentences, named entities, parts-of-speech (POS) and lemma. A \texttt{Doc} object represents one ICSR report and its corresponding free-text narratives as well as annotated spans. These \texttt{Doc} objects are then binarized and stored in a compressed \texttt{spacy} file format, one for each Train and Dev set.

Additionally, the variants’ tokenizers were responsible for converting the annotated spans into Beginning-Inside-Last-Outside-Unit (BILOU) tagging scheme. The hyperparameters of the parser remained consistent across all variants, with a hidden layer width of 64 and the state prediction layer consisting of 2 maxout pieces.

**Variant 1: Transition-Based CNN Model**

A "token-to-vector" (Tok2Vec) model was trained using pre-trained word vectors (en\_core\_web\_lg) and features extracted from free-text narratives. The Tok2Vec model itself is a Convolution Neural Network (CNN) with a Transition Based Parser (Parser) as its classification layer.
The `en_core_web_lg` word vectors were pre-trained on written texts from various blogs, news and comments. These texts were compiled to create 514,000 unique vectors, each with 300 dimensions.

The model Tok2Vec comprises of an embedding layer (`MultiHashEmbed.v2`) and an encoding layer (`MaxoutWindowEncoder.v2`). The embedding layer extracts subword features from the narratives, such as lemma, prefix, suffix and word "shape". These, along with pretrained vectors, are concatenated together and passed through a feed-forward neural network to form a mixed representation. The encoding layer employs a CNN (Section 2.4) with a maxout activation function [85].

Finally, to perform a structured prediction for named entities, a transition-based parser utilizes the network as a controller. This transition-based CNN model is provided by default in SpaCy and can be executed through a command line interface (CLI), which offers commands for training and tuning the models via a configuration file.

The base version of this model was trained on 256 input length (width = 256), with 8 convolutions layers (depth = 8) attending to 1 token on each side around a token (window size = 1), making it sensitive to $\text{depth} \times (\text{window} \times 2 + 1) = 24$ tokens at a time. The model was trained for 15 epochs with Adam optimizer at a learning rate of 0.001 and a stopping criterion (patience) of 1600 steps, which was implemented to halt training if the model’s loss did not minimize. The model was evaluated against the Dev set at the end of each epoch and results were logged in using WandB API on Weights and Biases platform [86].

To optimize the model’s performance, hyperparameter tuning was conducted using the Bayesian search method. A total of 33 model versions were trained and the best-performing model was saved. The most impacting hyperparameter configurations for all the models of this variant are captured in Appendix figure 6.7.

**Variant 2: Transition-Based RoBERTa Model**

A RoBERTa-BASE model (`spacy-transformers.TransformerModel.v3`) was employed for this setup, wherein the model was responsible for generating contextualized word embeddings, which served as inputs for the parser. The RoBERTa model had undergone pretraining on various English corpora, including Wikipedia articles and news sources. The model utilized cased vocabularies, meaning the words were not lower-cased before training and thus the model treated ‘English’ and ‘english’ differently. This version was specifically chosen as it could potentially understand the context in clinical narratives better, which most often times have drug and disease names capitalized.

This RoBERTa model accepts input sequences consisting of 512 tokens and operates with 768 hidden layers and 12 attention heads. In the case of narratives with more than 512 tokens, the input is divided before passing it through the transformer using a registered function (`spacy-transformers.strided_spans.v1`). To carry some context to this divided narrative, a context window of 128 token width is set with a stride of 96 tokens. During training, the model underwent 85 epochs, and an early stopping criteria (patience) of 1600 steps.

**Variant 3: Transition-Based Clinical BioBERT Model**

For this particular variant, the hyperparameter settings remained the same as those for the RoBERTa model, except that it was trained for a maximum of 100 epochs. The model itself was also replaced with Clinical BioBERT which is publicly available on HuggingFace. Like RoBERTa, this model also consists of 512 tokens as input sequence length, 768 hidden layers and 12 attention heads. It was initialized from BioBERT with cased vocabulary and further trained on MIMIC III clinical notes of a maximum sequence length of 128 tokens.
4.3 Stanza

Unlike MedspaCy pipelines, which handle the conversion of annotated spans into tokens with corresponding BILOU tags once presented with Doc object, Stanza, prefers annotations to be available in pretokenized Beginning-Inside-Outside (BIO) tagging scheme, to make use of its pre-built training pipelines. A custom NER creation required creating a clone of the Stanza project from GitHub and making changes to their core classes and functions in order to make the training pipeline recognize custom entities and the new dataset.

First, the tokenization was performed by calling Stanza’s tokenizer function. Subsequently, each token is checked for an exact match with the entity span. If a match is found, the token is given a Beginning ’B-<entity>’ tag (for example B-DRUG). If the entity is a multi-token entity, an Inside ’I-<entity>’ tag is also applied to represent the continuation of the entity. For the rest of the tokens, an Outside ’O’ tag is given. This also includes giving an ’O’ tag to tokens that are misaligned with entity spans. For instance, a case narrative with a span mismatch in “bone pain” like below:

Patient experienced bone pain two days after commencing Alvedon

will result in the following BIO tagging:

Patient experienced bone pain two days after commencing Alvedon

A custom function performing the above conversion is registered with Stanza core libraries and then the pipelines are ready to train on the new dataset with their custom entity tags using stanza.models.ner_tagger pipeline

Variant 1: RoBERTa embeddings ("Combined" vocabulary) with BiLSTM+CRF Model

The first variant of models constructed using Stanza comprised of a one-layer BiLSTM sequence tagger with a conditional random field (CRF) decoder. The model’s embeddings were a combination of word vectors trained using word2vec skipgram model, RoBERTa-BASE word embeddings and word embeddings from both forward and backward character-level LSTM model (CharLM) for capturing context on a character level.

The RoBERTa model used is the same one used in MedspaCy models. The word vectors obtained from the Word2Vec skipgram model are sourced from the OntoNotes [87] and CoNLL03 corpora [88] (a dataset created for a shared task on Computational Natural Language Learning), collectively referred to as the "combined" vocabulary and had a total of 250,000 word vectors of dimension size 100. The CharLM component was pretrained on the "Google One Billion Word" corpus [89] consisting of 0.8 million tokens.

The BiLSTM+CRF was trained for 20,000 steps with a batch size of 32. The training process employed Stochastic Gradient Descent (SGD) optimizer with an initial learning rate of 0.1, which gradually decayed with a factor of 0.5 if there was no improvement in training loss for 2 consecutive epochs. The model was evaluated on Dev set and logged in using WandB API at every 40 steps.

Unlike MedspaCy, which divides the narratives exceeding 512 tokens, Stanza took a different approach. It dropped reports from the training set that contained more than 512 tokens. Consequently, 40 out of 260 reports were removed from the Train set, and 11 out of 90 reports were removed from the Dev set.
Variant 2: RoBERTa embeddings (MIMIC vocabulary) with BiLSTM-CRF Model

For this variant, the model's architecture remained the same as in the previous variant, with one key difference. The word vectors obtained from word2vec and the weights of the CharLM were both replaced with word vectors and weights obtained after training specifically on the MIMIC-III dataset. The dataset consisted of 100,000 word vectors with a dimension size of 200.

For training, all the hyperparameters remained the same as variant 1, except patience was changed to 3 epochs, to let the model train for a longer time. Since the underlying RoBERTa model was the same, the tokenizer rejected the same count of reports which exceeded the maximum input token length as it did for variant 1, ie. 40 for the Train set and 11 for the Dev set.

Variant 3: Clinical BioBERT embeddings (MIMIC vocabulary) with BiLSTM-CRF Model

The last configuration setup in Stanza consisted of a clinical BioBERT model instead of RoBERTa-BASE. This model was also extracted from HuggingFace and therefore had the same architecture as the one used in MedspaCy. Like the previous variant, word vectors from MIMIC-III were employed but with pretrained clinical BioBERT.

With the architecture switched to BERT-BASE, the maximum token length of 512 affected the number of reports excluded from the dataset. In this variant, a total of 43 reports from the Train set of 260 and 13 reports from the Dev set of 90 were excluded due to exceeding the maximum token length. The training hyperparameters for the SGD optimizer remained the same as the last two variants.

4.4 CLAMP

CLAMP is available in two versions, with CLAMP-GUI being the only version providing us with clinical NLP pipelines for extracting features and training a new NER model. Since this version only accepts annotated clinical reports in XML(.xmi) file format, the Train/Dev/Test set had to be converted into one.

A sample of annotated file provided by CLAMP was analyzed to understand the structure in which each report in the dataset had to be converted to. The .xmi file was supposed to contain narrative text, sentence spans, token spans and their parts-of-speech (POS) and finally, entity spans with their names. The input transformation took place in four steps for each Train, Dev and Test set:

1. Each annotated JSON (.json) report was first converted into a plain text file (.txt) in order to manipulate them as per CLAMP XML standards.
2. An NLP pipeline (extract_features_AE_reports) was constructed in CLAMP which would split the sentences, tokenize the text and lastly add POS tags for each tokenized word.
   a) The DF_Clamp_sentence_detector component was used to split the sentences at "." (full-stop), "\n" (newline character) and sentences longer than 500 tokens. This component also takes care of full-stops occurring due to clinical or medical terminology by keeping a list of such terms and ignoring them, for example ".NO2" or "1 tabl./day".
   b) The rule-based tokenizer DF_Clamp_tokenizer component was employed for splitting tokens and saving their spans in the XML file.
   c) The DF_OpenNLP_POS_tagger component was used to attach POS tags for each identified token in the text and append this information in the XML file pertaining to that particular report.
3. All the reports in .txt format were copied into CLAMP and processed with this newly created NLP pipeline (extract_features_AE_reports). The pipeline returned an XML file for each processed input report file, which was later saved for further additions.
4. Lastly, the entity spans for each report were transferred from the JSON to the XML file. Special care was taken to address the escape characters and span shifts caused by the two
different file formats. All the escape characters had to be re-coded into XML accepted format (for example \r, \n carriage return and newline were converted to &\#13; and &\#10; respectively). Span shifts resulting from how XML handles "\n" were also handled appropriately.

Training
To train a new NER model with custom entities, a new corpus was created in CLAMP with Train and Dev sets loaded to their respective sub-folders. The tool allows the user to select the features to be incorporated for training with a drop-down menu and recommends selecting all 11 varieties of NER "feature extractor" components for the best performance. Post selection, the tool runs CRF classifier with Gradient Descent using L-BFGS method (CRFsuite component) on the extracted features and word vectors from the Train set. It is important to note that the hyperparameters of the CRFsuite cannot be customized and are preset to run with L-BFGS, with an L2 regularization coefficient of 0.9.

The "feature extractor" components (section 2.2.3) in CLAMP extract features from the unlabelled MIMIC-II dataset, which was provided during the SemEval 2014 challenge. When combined, these components generate a word vocabulary of 201,000 words and a total of 182 features.

4.5 Configuration Summary
All the model variants deployed by the three tools can be broadly summarised in the following three tables.

The table 4.1 compares the variants, the word embeddings they use, the model stack, its corresponding classification layer, the tagging scheme they follow and finally the maximum input sequence length allowed by them. A '-' symbol under "Pretrained word embedding" represent an absence of any word vectors used apart from what the pretrained model already comes packed with. A '-' symbol under "Seq. length" means that the field is not applicable to that tool’s configuration.

<table>
<thead>
<tr>
<th>NLP tool</th>
<th>Pretrained Word Emb.</th>
<th>Model</th>
<th>Classification layer</th>
<th>Tagging scheme</th>
<th>Seq. length</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedspaCy variant 1</td>
<td>en_core_web_lg</td>
<td>Tok2Vec (CNN)</td>
<td>Transition based parser</td>
<td>BILOU</td>
<td>256</td>
</tr>
<tr>
<td>MedspaCy variant 2</td>
<td>-</td>
<td>RoBERTa</td>
<td>Transition based parser</td>
<td>BILOU</td>
<td>512</td>
</tr>
<tr>
<td>MedspaCy variant 3</td>
<td>-</td>
<td>Clinical BioBERT</td>
<td>Transition based parser</td>
<td>BILOU</td>
<td>512</td>
</tr>
<tr>
<td>CLAMP</td>
<td>MIMIC</td>
<td>CRFSuite</td>
<td>CRF</td>
<td>BIOES</td>
<td>512</td>
</tr>
<tr>
<td>Stanza variant 1</td>
<td>Combined</td>
<td>RoBERTa + CharLM(1billion) + Bi-LSTM</td>
<td>CRF</td>
<td>BIOES</td>
<td>512</td>
</tr>
<tr>
<td>Stanza variant 2</td>
<td>MIMIC</td>
<td>RoBERTa + CharLM(MIMIC) + Bi-LSTM</td>
<td>CRF</td>
<td>BIOES</td>
<td>512</td>
</tr>
<tr>
<td>Stanza variant 3</td>
<td>MIMIC</td>
<td>Clinical BioBERT + CharLM(MIMIC) + Bi-LSTM</td>
<td>CRF</td>
<td>BIOES</td>
<td>512</td>
</tr>
</tbody>
</table>

Table 4.1. Variant configuration comparisons

Table 4.2 is a deeper look into the word embeddings/vectors or feature vectors deployed by each tool and the sources/ vocabularies on which they were originally trained on. The "Model" column lists the neural networks or the algorithms used to generate the corresponding vectors/features. In the case of CLAMP and Stanza, these results were derived either from the log files generated by the tool or from its published paper. In the case of MedspaCy, this information was sourced from its official documentation. All the values under "Emb./vocab size" are rounded off to their nearest
thousands.

<table>
<thead>
<tr>
<th>NLP tool</th>
<th>Pretrained Word Emb. / features</th>
<th>Model</th>
<th>Vocabulary / sources</th>
<th>Emb. / feature Dim.</th>
<th>Emb. / vocab size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedspaCy</td>
<td>en_core_web-lg</td>
<td>fastText + Bloom emb.</td>
<td>OntoNotes 5, ClearNLP, WordNet 3.0, Explosion Vectors</td>
<td>300</td>
<td>514k</td>
</tr>
<tr>
<td>CLAMP</td>
<td>MIMIC</td>
<td>Brown clustering, Rand index clustering, K-means, UMLS lookup</td>
<td>MIMIC - II</td>
<td>182</td>
<td>201k</td>
</tr>
<tr>
<td>stanza</td>
<td>Combined Word2vec skipgram</td>
<td>OntoNotes, CoNLL03</td>
<td>MIMIC-III</td>
<td>100</td>
<td>250k</td>
</tr>
<tr>
<td>stanza</td>
<td>MIMIC</td>
<td>Word2vec skipgram</td>
<td>MIMIC-III</td>
<td>200</td>
<td>100k</td>
</tr>
</tbody>
</table>

Table 4.2. Pretrained word vectors / embeddings and their vocabularies

Lastly, the table 4.3 lists the pretrained models that were further fine-tuned in all the variants for the NER task. It also lists the corpora on which these models were first trained. It is to be noted that the clinical BioBERT, unlike the rest of the models that were trained from scratch, was initialized from a pretrained BioBERT model and fine-tuned on the MIMIC-III dataset. A ‘-’ symbol represents that the corresponding model is not storing any pretrained weights.

<table>
<thead>
<tr>
<th>NLP tool</th>
<th>Model</th>
<th>Corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedspaCy variant 1</td>
<td>Toke2Vec (CNN)</td>
<td>-</td>
</tr>
<tr>
<td>MedspaCy variant 2</td>
<td>RoBERTa</td>
<td>BookCorpus, English Wikipedia, CC-New, OpenWebText, Stories (CommonCrawl)</td>
</tr>
<tr>
<td>MedspaCy variant 3</td>
<td>Clinical BioBERT</td>
<td>MIMIC -III</td>
</tr>
<tr>
<td>CLAMP</td>
<td>CRFSuite</td>
<td>-</td>
</tr>
<tr>
<td>Stanza variant 1 &amp; 2</td>
<td>RoBERTa</td>
<td>BookCorpus, English Wikipedia, CC-New, OpenWebText, Stories (CommonCrawl)</td>
</tr>
<tr>
<td>Stanza variant 3</td>
<td>Clinical BioBERT</td>
<td>MIMIC -III</td>
</tr>
<tr>
<td>Stanza variant 1</td>
<td>CharLM (1billion)</td>
<td>Google One Billion Word</td>
</tr>
<tr>
<td>Stanza variant 2 &amp; 3</td>
<td>CharLM (MIMIC)</td>
<td>MIMIC-III</td>
</tr>
</tbody>
</table>

Table 4.3. Pretrained models and their sources

4.6 Evaluation

All three tools have their own internal scoring mechanisms for evaluating their trained models. While MedspaCy and Stanza come with a Python class and CLI command respectively, CLAMP does not provide any option to run the trained models on a Dev/Test set for evaluation purposes. It, however, gives the evaluation scores on Dev/Test set after a new model has just been trained. After going through each tool’s codebase, the following observations were made on how they evaluate the Dev/Test sets:
1. MedspaCy and Stanza report micro-Precision(P), micro-Recall(R) and micro-F1 scores, while CLAMP provides a macro-averaged score on all three metrics. Only Stanza provides a confusion matrix for each entity type as well as for tokens classified with an 'O' (Outside) tag.

2. MedspaCy has two types of P/R/F1 scores: one on a global level and the other on an entity level, i.e., by performing an exact match of both spans and the entity type. Stanza provides only global level P/R/F1 scores but in two forms: calculation done by counting entity types in ground truth data and predictions, and calculation done by just matching the token classes without matching the span information. Lastly, CLAMP also provides global-level P/R/F1 scores.

Owing to such differences in scoring mechanisms across all tools, there was a need to come up with an in-house evaluation mechanism to have a fair comparison.

The in-house scoring mechanism performed an exact match of spans and entity types of the model’s prediction with the ground truth data and reports global as well as entity-level P/R/F1 micro-averaged scores. For instance, for the narrative text "Patient experienced bone pain two days after commencing Alvedon" the following will be the result of the scoring mechanism:

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>Predictions</th>
<th>Exact Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Span</td>
<td>End Span</td>
<td>Entity</td>
</tr>
<tr>
<td>22</td>
<td>29</td>
<td>AE</td>
</tr>
<tr>
<td>30</td>
<td>39</td>
<td>DATE</td>
</tr>
<tr>
<td>57</td>
<td>64</td>
<td>DRUG</td>
</tr>
</tbody>
</table>

Note, a way to calculate P/R/F1 scores is by inferring them from a confusion matrix. However, since such a matrix is challenging to come up without tokenizing the narrative first and the fact that each tool uses its own tokenization method, the calculation of scores was not performed based on the matrix. For the same reason, a relaxed match (instead of an exact match) of spans and entity types could not be implemented. Rather, a simple criteria of checking exact matches of spans and entity types was adopted. Therefore, if (span start, span end, entity) represents a tuple in the ground truth(Gold) and prediction(Pred) sets, the following can be said about true positive(TP), false positives(FP) and false negatives(FN):

\[
TP = \text{Pred} \cap \text{Gold} \quad (4.1)
\]
\[
FP = \text{Pred} \setminus \text{Gold} 
\quad (4.2)
\]
\[
FN = \text{Gold} \setminus \text{Pred} 
\quad (4.3)
\]
\[
\text{Gold, Pred } \ni (\text{span start, span end, entity})
\]

The predictions from the three tools were first prepared in a standard format of (span start, span end, entity) tuples and then evaluated against the ground truth tuples. The span information were derived from Document object in MedspaCy and Stanza, while for CLAMP, the XML output was parsed to extract the relevant span information. This was done for each ICSR report in the Dev and Test set and final global and entity level scores were reported.
5. Results

5.1 NER Performance

For the comparison among different NLP tools, the results (table 5.1) were either reported from the inherent NER models or from the publicly available HuggingFace models. MedspaCy variant 2 consisting of a pretrained RoBERTa model with transition based parser and F1 score of 81.98%, outperformed the next best MedspaCy variant 3, pretrained Clinical BioBERT model, by a \( \Delta \) F1 score of +0.63%. The worst performing was reported to be MedspaCy’s variant 1 which was trained on Tok2Vec model with pretrained word embeddings.

The top 2 overall performers were in fact from MedspaCy followed by all the variants from Stanza and then by CLAMP, using the in-house metric. This was different from what the internal metrics were revealing, where the top 2 positions were earlier reported to be Stanza variants.

<table>
<thead>
<tr>
<th>NLP Tool</th>
<th>Internal scorer</th>
<th>In-house scorer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>MedspaCy variant 1</td>
<td>76.25</td>
<td>73.7</td>
</tr>
<tr>
<td>MedspaCy variant 2</td>
<td>84.18</td>
<td>81.66</td>
</tr>
<tr>
<td>MedspaCy variant 3</td>
<td>83.45</td>
<td>80.69</td>
</tr>
<tr>
<td>CLAMP</td>
<td>80.12*</td>
<td>64.08*</td>
</tr>
<tr>
<td>Stanza variant 1</td>
<td>87.92</td>
<td>80.37</td>
</tr>
<tr>
<td>Stanza variant 2</td>
<td>86.83</td>
<td>81.92</td>
</tr>
<tr>
<td>Stanza variant 3</td>
<td>85.23</td>
<td>76.98</td>
</tr>
</tbody>
</table>

**Table 5.1.** Precision (P), Recall (R) and F1 scores for Test dataset by tool’s internal scorer vs. in-house scorer. All values reported are micro-averaged, except for CLAMP, which is macro-averaged and marked with (*). All values are in percentage. **Bold** scores are best overall in that scorer category. Underlined scores are best for the NLP tool’s category.

For selecting a model for production at UMC, based on the precision and recall scores, it is preferred to have a higher recall score with less compromise in precision, indicating that the model effectively captures the presence of entities in the narrative and covers most of the ground truth. Figure 5.1 shows that the variant with highest recall, which also happens to be the highest scorer, is Medspacy variant 2.

5.2 Speed Comparisons

The training times of various NLP tools were compared to the fastest performing one (table 5.2). CLAMP was able to train its CRF-based model using L-BFGS optimizer in just 3 minutes, whereas the Stanza variants took an average of 15 hours with SGD optimizer. However, the training for the Stanza variants was halted as there was not much improvement in performance observed. MedspaCy, on the other hand, trained its Tok2Vec model for 48 minutes and transformer based models for 9 hours on an average with Adam optimizer. If a GPU device had been available during training, both the MedspaCy and Stanza models could have utilized GPU cores, leading to decreased training times.
Figure 5.1. Precision-Recall curve and iso-F1 curves (red).

<table>
<thead>
<tr>
<th>Task</th>
<th>MedspaCy Tok2Vec</th>
<th>MedspaCy Transformer</th>
<th>CLAMP</th>
<th>Stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER training</td>
<td>16×</td>
<td>177×</td>
<td>1×</td>
<td>300×</td>
</tr>
</tbody>
</table>

Table 5.2. Average runtime (CPU) of various tools relative to CLAMP on Train set. Since MedspaCy uses a non-transformer (Tok2Vec) model variant, it has been separated into its own column.

5.3 User Interface

A user interface (UI) was created using Python Streamlit package, adhering to the design principles of UMC’s other research tools and considering the interests of PV scientists. This UI was an attempt to provide PV assessors with the experience of the best model in production. Moreover, it potentially paves the way for future evaluations of the NER prototype tool.

Integrating the UI and the backend model, the NER prototype tool as a whole enables users to enter up to five report IDs. It retrieves the reports from VigiBase, processes the narratives using the NER model, and presents the extracted entity spans and types for each report on the UI. The user then has an option to either see all the entities highlighted in narratives or only the entities of their choice. Additionally, an alternative option is provided to enter free-text narratives in a text box for detecting entity spans. The UI also hyperlinks all the reports to UMC’s signal detection and signal management tool, VigiLyze, for retrieving other relevant information related to the case.
5.4 Discussions

NER Performance

MedspaCy variants captured 4 out of 5 entities with the highest F1 scores (table 5.3), namely AE, DRUG, NEGATION and PROBLEM. Meanwhile Stanza variants captured DATE, the best. CLAMP did not score higher than any other tool’s variant although it had the 2nd highest score in DRUG entity detection among the tools. A general trend seen in all the clinical tools is that they give high scores for DATE and NEGATION among all the entities. The observed order of increasing difficulty in entity recognition was as follows: DATE < NEGATION < DRUG < AE < PROBLEM.

Without a confusion matrix, it is difficult to comment on whether the low scores on certain entities especially PROBLEM, were a cause of misclassification or unclassification to any entity by the model. Intuition says that it’s highly likely for the models to confuse PROBLEM as an AE entity, therefore a case of misclassification. But the tables 3.4 & 6.1 (in Appendix) give a picture of high false negatives, leading to very low recall across all models and hence a case of unclassification, given the fact that false positives for AE remain low i.e., precision remains high.

<table>
<thead>
<tr>
<th>NLP Tool</th>
<th>AE</th>
<th>DRUG</th>
<th>NEGATION</th>
<th>DATE</th>
<th>PROBLEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedspaCy variant 1</td>
<td>62.11</td>
<td>67.85</td>
<td>64.85</td>
<td>90.54</td>
<td>51.56</td>
</tr>
<tr>
<td>MedspaCy variant 2</td>
<td>79.56</td>
<td>76.96</td>
<td>78.24</td>
<td>92.83</td>
<td>69.11</td>
</tr>
<tr>
<td>MedspaCy variant 3</td>
<td>78.11</td>
<td>71.55</td>
<td>74.68</td>
<td>90.66</td>
<td>53.03</td>
</tr>
<tr>
<td>CLAMP</td>
<td>75.79</td>
<td>54.76</td>
<td>63.58</td>
<td>90.56</td>
<td>53.03</td>
</tr>
<tr>
<td>Stanza variant 1</td>
<td>77.73</td>
<td>72.97</td>
<td>75.28</td>
<td>90.38</td>
<td>72.62</td>
</tr>
<tr>
<td>Stanza variant 2</td>
<td>75.92</td>
<td>76.25</td>
<td>76.08</td>
<td>91.9</td>
<td>69.84</td>
</tr>
<tr>
<td>Stanza variant 3</td>
<td>76.28</td>
<td>67.71</td>
<td>71.77</td>
<td>90.43</td>
<td>68.42</td>
</tr>
</tbody>
</table>

Table 5.3. Precision (P), Recall(R) and F1 scores per Entity type for Test dataset. All values reported are micro-averaged and in percentage. Bold scores are best overall in the entity category. Underlined scores are best for the NLP tool’s category.

After analyzing the three tools, it was observed that they handle the count of input tokens differently post-tokenization. CLAMP does not have an upper limit to the count it can process, but MedspaCy and Stanza have limitations due to the nature of their models. MedspaCy uses...
both CNN and transformer based models, as a result, it cuts long documents into small sequences post-tokenization. The new documents created have some overlapping tokens from the previous document to ensure that they carry some left and right context. Stanza, on the other hand, drops such a document completely and does not process it during training time. Thus, to improve the training of Stanza’s model, the longer reports need to be broken down into shorter ones in the preprocessing step.

The Test set and the in-house scorer for evaluation reveal a few interesting points of discussion too. First, from table 5.4, it can be observed that Test scores are higher than Dev scores in both internal and in-house scoring metrics. Secondly, in-house scores for both Dev and Test sets are harsher than internal scores, except for CLAMP, where the scores are higher.

<table>
<thead>
<tr>
<th>NLP Tool</th>
<th>Model</th>
<th>Internal scorer</th>
<th>In-house scorer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedspaCy variant 1</td>
<td>Tok2Vec(CNN) + TBP</td>
<td>72.83 74.95</td>
<td>71.54 73.91</td>
</tr>
<tr>
<td>MedspaCy variant 2</td>
<td>RoBERTa + TBP</td>
<td>78.77 82.9</td>
<td>77.38 81.98</td>
</tr>
<tr>
<td>MedspaCy variant 3</td>
<td>Clinical BioBERT + TBP</td>
<td>76.88 82.05</td>
<td>75.75 81.35</td>
</tr>
<tr>
<td>CLAMP</td>
<td>CRFSuite</td>
<td>68.94* 70.5*</td>
<td>72.81 76.14</td>
</tr>
<tr>
<td>Stanza variant 1</td>
<td>RoBERTa + CharLM(1billion) + Bi-LSTM + CRF</td>
<td>79.55 83.98</td>
<td>77.17 80.76</td>
</tr>
<tr>
<td>Stanza variant 2</td>
<td>RoBERTa + CharLM(MIMIC) + Bi-LSTM + CRF</td>
<td>79.63 84.31</td>
<td>76.42 79.74</td>
</tr>
<tr>
<td>Stanza variant 3</td>
<td>Clinical BioBERT + CharLM(MIMIC) + Bi-LSTM + CRF</td>
<td>78.63 80.89</td>
<td>74.93 78.88</td>
</tr>
</tbody>
</table>

Table 5.4. F1 scores for Dev and Test datasets by tool’s internal scorer vs. in-house scorer. All values reported are micro-averaged, except for CLAMP, which is macro-averaged and marked with (*). All values are in percentage. Bold scores are best overall in that scorer category. Underlined scores are best for the NLP tool’s category.

Tools receiving higher Test scores than Dev scores can be explained as follows:

- Test set was cut off from the last 100 out of 450 annotated reports. Even though the samples were randomly selected, as per table 3.4, the Test set received much fewer PROBLEM entities, even though the Test set had 100 reports in comparison with Dev set with 90 reports. Since PROBLEM entity is known to be the most difficult to capture, having fewer of them to classify in Test set than Dev set, increases all the tools’ scores. It is to be noted that Test set was created after Train and Dev split was already performed and NER model exploration on MedspaCy had already begun.
- Since Test set was created afterward, it is possible that the author’s knowledge of identifying trickier entities (3.2.3) might have improved the annotation quality. A proof of this can be derived from the confusion matrix reported by Stanza variant 1 for Dev and Test set (table 6.2 & 6.3). The PROBLEM entity misclassified as ‘O’ (Outside entity) in Test set is less than Dev set, even though the count of reports are more.

In-house scorer metrics are more conservative than internal scorer metrics because of the following reasons concluded after comparing code-bases:

- The MedspaCy scorer excludes Documents where the model failed to predict any entity from evaluation. The in-house scorer includes such cases during evaluation and therefore the overall false negatives increase, decreasing recall for the model. Similarly, MedspaCy scorer excludes Documents with no annotations in them (6 reports in the 450 annotated set).
from evaluation. If included, these reports will increase the false positive rate, decreasing the precision of the model.

- The Stanza scorer, unlike the in-house scorer, does not perform an exact match of prediction and gold standard annotation. It rather ignores the span information, only checks the entity name and matches the order in which they appeared in the prediction and gold annotation set. This means that true positive cases are reported higher and false positive/negative cases are reported lower by the Stanza scorer.
- It was difficult to assess CLAMP codebase for its scoring mechanism, since it is in binarized format, and determine the reason it reported high scores with in-house scorer.

**Effects of using pretrained word embeddings**

Different word vectors and embeddings were utilized in the models, including domain-specific ones trained on MIMIC-II and MIMIC-III datasets, as well as those trained on general English corpora. While no specific ablation study was conducted to isolate their impact on each variant’s performance, it can be observed that their inclusion did not lead to an improvement. In fact, there was a decrease in the F1 score of 1.02% in Stanza variant 2 compared to variant 1. This suggests that despite ICSR reports being clinical in nature, the diverse content within these reports indicates that a NER model would benefit more from word vectors and embeddings trained on general English corpora.

**Effects of using models pretrained in clinical domain vs. general domain**

Another interesting outcome of using domain-specific model and embeddings was that the clinical BioBERT model performed worse than RoBERTa model in both MedspaCy and Stanza variants. This goes against the prevalent assumption that a domain-specific pretrained model and embeddings will always perform better. There can be various reasons due to which the clinical BioBERT model did not pan out well with free-text narratives in VigiBase for NER task:

- The MIMIC-III dataset, which served as the corpus used in pretraining this model, exclusively consists of notes from the intensive care unit of a single healthcare institution. However, VigiBase holds data coming from various reporter types, institutions and countries which follow different reporting practices. For such a diverse dataset, a model pre-trained solely on data from a single institution might not be very effective.
- The model was also pretrained with a sequence length of 128 tokens versus roBERTa which saw 512 tokens while pretraining. This could have affected the pretrained weights and embeddings, as they were only able to see a shorter context. The narratives used in our case had long texts and entities like PROBLEM and AE were referred in each other’s context over long sequences.
- The data quality and annotation inconsistency can also affect the performance. The embeddings derived from a general English corpus tend to be larger and more diverse, potentially impacting the model’s ability to generalize effectively. Moreover, annotation inconsistency in our dataset could have made it difficult for the clinical model to generalize as it may have primarily encountered consistent annotations during its pretraining phase. In contrast, the RoBERTa model likely encountered a wider range of annotation inconsistencies from various datasets while being pretrained.

**System Usage**

Both MedspaCy and Stanza provide Python packages and CLI commands to perform various NLP tasks relevant to NER model building, such as tokenization, sentence segmentation, adding pretrained word vectors, adding pretrained language models and logging the model outputs and evaluation pipelines. Their library architectures also follow closely except MedspaCy, which is
built on SpaCy and offers highly modular and customizable pipelines. They both also provide modules to access each other’s tokenizer, segmenter and models. CLAMP has a standalone Java application to perform all the mentioned NLP tasks but with a limited set of components and is impossible to integrate third-party models and modules.

MedspaCy, because of its customizable modules and attention to making the pipelines run fast (table 5.2) and efficiently, has made it easier for a user to develop a model but at the same time making it relatively harder to understand the codebase. Stanza, on the other hand, due to its less customizable nature, claims to develop more accurate models.
6. Conclusion

In order to address the question of how can the existing information extraction (IE) tools be fined-tuned to support signal assessment of adverse event report narratives, three NLP tools were explored and their models were trained to detect named entities in free-texts of VigiBase reports. This work proved that it is possible to fine-tune existing IE tools to support signal assessment. A user interface was also developed in an attempt to measure the impact of IE tool on the performance of PV assessors and their experience with the tool.

To develop the IE tool, a dataset was extracted from VigiBase and annotated, consisting of various ICSRs. It was then canonically split based on which NER models were trained. For building these models, MedspaCy, CLAMP and Stanza NLP tools and their pipelines were utilized. A user interface to these NER models processed ICSR narratives and visualized five named entities: Adverse Events (AE), Drug, Negation, Date and Problem. The overall contribution of the author includes the annotation guideline, annotated datasets, model configurations, word embeddings and model training and evaluation scripts. The best-performing model with 81.98% F1 score was noted to be MedspaCy variant 2, which ran RoBERTa model to generate word embeddings and had a transition-based parser as the classifier.

6.1 Limitations and Future Work

During the project lifecycle, the following notable limitations were identified, which present opportunities for future work and improvement:

- During the report exclusion step in dataset preparation, free-texts from patient history, sender comments and reporter comments sections were not checked for the presence of the English language. It is highly possible that the embeddings thus produced by NER models were getting polluted with non-English words. It was also observed that the case narratives contained personal health information (PHI) in the form of de-identified words such as ’XX’ and abbreviations. These masked words could have also polluted the word embeddings.
- Annotations were performed by the author, who is not a subject expert and would have thus produced inconsistent annotations across the dataset. Since having a quality labeled dataset is essential in a supervised learning task, it is unquestionable to have the dataset annotated by experts and build the process stronger by having an inter-annotator agreement. If an expert annotator is unavailable, other methods [91] of using noisy annotation for NER task should be taken into consideration.
- Most of the successful NER systems depend on highly engineered hand-crafted rules and dictionary lookups for text processing, which were not explored in this work. Since, all three NLP tools used provide a number of pipelines to build these custom rules, implementing them could have potentially improved their model’s performance. As a future work, the NER pipelines developed in this thesis should be considered to be a baseline and further text processing rules should be explored to better suit the dataset.
- It was observed that pretrained word embeddings from clinical domains (MIMIC dataset) did not perform better than the ones trained on general domain (such as English Wikipedia). The behavior of both kinds of word embeddings should be further investigated in order to take full advantage of them. Word vectors can also be built separately using all VigiBase reports and can then be used in addition to word embeddings from general domain datasets to fine-tune NER models.
• Finally, the evaluation of the overall user experience with the IE tool and its impact on PV assessor’s performance using suitable metrics is left as a scope for future work.

Apart from the aforementioned limitations and future work, this thesis work can be of immense use if integrated into a tool that could benefit from an NER model. An Entity Linking (EL) task can utilize a NER model where relationships among the entities can be identified and these entities can then be linked to UMC’s knowledge base for fetching relevant documents. The NER model can also be employed along with generative language models where it can provide better context in order to generate results to prompt queries. Lastly, the NER model can facilitate the annotation of more narratives through active learning by overcoming cold-start.


[99] K. Buchan, M. Bari, and A. Stubbs. Annotation guidelines for the adverse drug event (ADE) and medication extraction challenge. 2018. URL: n2c2.dbmi.hms.harvard.edu/files/ADE%20%5C_Annotation%5C_Guideline%5C_final.pdf.

Appendix A.
Annotation Guidelines

Overview
The aim is to create a quality annotated dataset for training and testing of a NER model. This task includes identifying the following tags:

- Drug
- Adverse Event (AE)
- Problem
- Date
- Negation

The tags used to indicate the presence of drug information are in accordance with those found in free-text narratives of Individual case safety reports (ICSR) in VigiBase which capture the Adverse Drug Reactions (ADR) on patients taking those medications. This document provides a high-level overview of annotations, how each tag is defined and the reasoning that goes behind annotating or not annotating an entity. The document is not intended to provide a complete justification of these reasons, but specific annotation decisions through examples.

The annotation is performed on a dataset extracted from VigiBase consisting of case narratives, reporter's comments, patient's medical history and sender comments.

Background and definitions
In their report 'International Drug Monitoring: The Role of National Centres’ from 1972, The World Health Organization (WHO) defines an Adverse Drug Event (ADE) as: "an injury resulting from medical intervention related to a drug" [94]. Where the term "injury" [95] may or may not have a causal relationship with the administration of this drug. It can be an unintended sign (including an abnormal laboratory finding), symptom, or disease temporally associated with the use of this drug.

For the purpose of this project, Council For International Organizations Of Medical Sciences (CIOMS) definition of Adverse Event will be used, since it is a broader term that encompasses the definition of ADE given by WHO.

The CIOMS definition of AE is: "Any untoward medical occurrence in a patient or clinical trial subject administered a medicinal product and which does not necessarily have a causal relationship with this treatment."[96]

A Drug is a synonym to medicinal product / medicine for our annotation purposes. It is defined by EU Directive 2001/83/EC [97] as 1. "Any substance or combination of substances presented for treating or preventing disease in human beings." 2. "Any substance or combination of substances which may be administered to human beings with a view to making a medical diagnosis or to restoring, correcting or modifying physiological functions in human beings is likewise considered a medicinal product."

Tags
This section talks about the entities which we are interested in capturing. Yellow highlighted are the correct annotations and the underlines terms/phrases in red are wrong annotations. The premise for all the below example is that the patient is experiencing muscle pain, secondary to statin therapy for coronary artery disease.
Drug
This tag captures a specific drug or a class of drugs.

- Drugs suspected of causing ADE, suspected of interacting with another drug to cause ADE or concomitant to drugs causing ADE should be annotated.
- Drugs that are negated in the narrative should be annotated.
- Herbals and traditional medicines should be annotated.
- Illicit drugs and alcohol should not be annotated.

For example:
1. Patient is experiencing muscle pain, secondary to statin therapy for coronary artery disease. [6]
2. The patient suffers from steroid-induced hyperglycemia. Patient prescribed 1 x 20 mg Prednisone tablet daily for 5 days. [99]
3. Patient has been switched to Lisinopril 10mg 1 tablet PO QD. [6]

AE
This tag captures an Adverse Event

- Annotate all AE in "chain of events". Do not establish causal relations in the chains: annotate each AE separately. For example, in "Sepsis followed by shock", annotate sepsis alone, shock alone, and NOT sepsis + shock. [98]
- Lab test results and findings should be annotated.
- Withdrawal symptoms should be annotated as they are caused by the drug indirectly.
- Diseases or medical conditions/events present before the drug was administered should not be annotated.

For example:
1. Rhabdomyolysis resulting in acute prerenal failure and long-term dialysis has been associated with Livalo. [98]
3. Patient received 100 Units/kg IV heparin sodium injection for treatment of deep vein thrombosis. [6]

Problem
This tag captures current and historical indications for medical treatments or diseases the patient has been or is suffering from.

- Annotate all the diseases the patient suffered from (patient history) before the drug was administered.

For example:
2. Patient received 100 Units/kg IV heparin sodium injection for treatment of deep vein thrombosis. [6]

Date
This tag captures the different forms date and time can take while describing the case narrative.

- The duration for which the drug was administered should be annotated, including the time to onset.
- Times of (the) day should be annotated. Morning, afternoon etc.
- Drug frequency should not be annotated. Every day, twice a day etc.
• The age and date of birth (DOB) of a patient should not be annotated. Since DOB is a personal identifier such cases should be reported back to the VigiBase team separately in order to be removed from dataset.

For example:
1. Patient prescribed 1 x 20mg Prednisone tablet daily for 5 days [6]
2. 5 hours after starting on the drug, the patient developed rashes all over the body.
3. Prescribed drug was taken on the morning of reported reaction. Exact time of ingestion is unknown.
4. On 20/01/2010, the patient developed rashes all over the body.

Negation
This tag captures the negation triggers of ADR or drugs.
• Negation triggers before and after the ADR or drugs should be annotated.
• Some common negation triggers that should be annotated: 'unknown', 'uncertain', 'not', 'no', 'without', 'ruled out', 'unlikely', 'absence of', 'denied', 'negative for' etc. [95] Note: entities showing incomplete negation or uncertainties like 'unknown', 'uncertain' are also annotated.
• A double negation is still considered a negation depending on the context and so should be annotated. Example: "not absent" is considered a negation trigger.

For example:
1. extremities showed no cyanosis, clubbing, or edema [95]
2. The test result for Covid-19 report was negative.
3. The viral infection was absent.

FAQs

**Question:** Should IVF be annotated as a drug? Any other forms like PRBCs?

**Answer:** IVF and PBCs should be annotated as Drugs. In vitro fertilisation (IVF) is one of several techniques available to help people with fertility problems have a baby. If hormones used in IVF are mentioned, they are drugs. Packed red blood cells (PRBCs) are made from a unit of whole blood by centrifugation and removal of most of the plasma, leaving a unit with a hematocrit of about 60%. It is on the World Health Organization’s List of Essential Medicines and therefore considered drugs.

**Question:** What about placebo drugs?

**Answer:** If the "placebo drug" is a medicinal product, then any effect that is adverse, should be annotated. The placebo drug should be annotated as Drug and the adverse event caused by any medicinal product (placebo drug or otherwise) leading to placebo effect reported as adverse should be annotated as ADR. Source of such drugs can be found in [100].

**Question:** Should Umbrella terms like 'Analgesics' or 'Pain killer' annotated? What about herbals and traditional medicine, for example "gingko"?

**Answer:** Both umbrella terms and herbal/traditional medicines should be annotated as Drug entity since they fall under the UMC definition of drug.
Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Designation</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alem Zekarias</td>
<td>Senior Pharmacovigilance Scientist, PV Science</td>
<td>Adjudicator</td>
</tr>
<tr>
<td>Eva-Lisa Meldau</td>
<td>Senior Data Scientist, Data Science</td>
<td>Project Co-Supervisor</td>
</tr>
<tr>
<td>Jayant Yadav</td>
<td>Master Thesis Student, Data Science</td>
<td>Annotator, Author</td>
</tr>
<tr>
<td>Joana Félix</td>
<td>Medical Informatician, PV Science</td>
<td>Adjudicator</td>
</tr>
<tr>
<td>Joana Rosado Coelho</td>
<td>Head of Data Science, Data Science</td>
<td>Project Supervisor</td>
</tr>
<tr>
<td>Qun-Ying Yue</td>
<td>Senior Pharmacovigilance Expert, Signal Management</td>
<td>Adjudicator</td>
</tr>
</tbody>
</table>

Change Log

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>02/02/2023</td>
<td>1.0</td>
<td>First Draft</td>
</tr>
<tr>
<td>06/02/2023</td>
<td>1.1</td>
<td>Grammar corrections, 'Monitoring for ADR' point dropped.</td>
</tr>
<tr>
<td>09/02/2023</td>
<td>2.0</td>
<td>Updated ADE definition based on CIOMS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Withdrawal symptoms addressed in ADR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Placebo effect and drugs addressed in FAQs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOB addressed in ADR Tags.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Double negation addressed under Negation Tags.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defined a new entity &quot;Problem&quot; for annotation.</td>
</tr>
<tr>
<td>16/02/2023</td>
<td>2.1</td>
<td>Added AE definition based on CIOMS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scrapped WHO and UMC definitions of ADR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Herbal medicines and umbrella terms addressed in FAQs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refined the concept of negation triggers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated definition of Drug.</td>
</tr>
<tr>
<td>28/02/2023</td>
<td>2.2</td>
<td>Herbal medicines addressed in Drug tag.</td>
</tr>
<tr>
<td>04/03/2023</td>
<td>2.3</td>
<td>Due to a lack of references of UMC definition of AE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the definitions are replaced by the one mentioned in CIOMS.</td>
</tr>
<tr>
<td>10/03/2023</td>
<td>2.4</td>
<td>Due to a lack of references of UMC definition of AE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the definitions are replaced by the one mentioned in Directive 2001/83/EC.</td>
</tr>
</tbody>
</table>

63
Appendix B.
Figures

Figure 6.1. MedspaCy library architecture [42]
Figure 6.2. Percentage of reports per feature type

Figure 6.3. Lifecycle of annotation process
Figure 6.4. MedspaCy evaluation curves for all variants. Left: Training loss for Train set. Right: F1 score for Dev set. 1 Step represents 0.5 epochs for variant 1 and 2.5 epochs for variant 2&3.

Figure 6.5. CLAMP training loss for Train set.

Figure 6.6. Stanza evaluation curves for all variants. Left: Training loss for Train set. Right: F1 score for Dev set.
Figure 6.7. Hyperparameter configurations for a few models of variant 1 and all models of variant 2&3 that were trained in MedspaCy. Only those hyperparameters which had the most impact on the scores are shown. Configurations having <null> in HF model column means, spaCy’s internal model ie., *Tok2Vec* was used instead of HuggingFace. <null> in Learning Rate column means, that it was not fixed throughout the training process but used warmup steps starting from $5 \times 10^{-5}$.

Figure 6.8. Hyperparameter configurations for all models trained in Stanza. Only those hyperparameters which had the most impact on the scores are shown.
### Appendix C.

#### Tables

<table>
<thead>
<tr>
<th>NLP Tool</th>
<th>AE</th>
<th>DRUG</th>
<th>NEGATION</th>
<th>DATE</th>
<th>PROBLEM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP</td>
<td>FP</td>
<td>FN</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>MedspaCy variant 1</td>
<td>477</td>
<td>291</td>
<td>226</td>
<td>392</td>
<td>109</td>
</tr>
<tr>
<td>MedspaCy variant 2</td>
<td>541</td>
<td>139</td>
<td>162</td>
<td>464</td>
<td>60</td>
</tr>
<tr>
<td>MedspaCy variant 3</td>
<td>503</td>
<td>141</td>
<td>200</td>
<td>470</td>
<td>80</td>
</tr>
<tr>
<td>CLAMP</td>
<td>385</td>
<td>123</td>
<td>318</td>
<td>426</td>
<td>47</td>
</tr>
<tr>
<td>Stanza variant 1</td>
<td>513</td>
<td>147</td>
<td>190</td>
<td>453</td>
<td>110</td>
</tr>
<tr>
<td>Stanza variant 2</td>
<td>536</td>
<td>170</td>
<td>167</td>
<td>435</td>
<td>83</td>
</tr>
<tr>
<td>Stanza variant 3</td>
<td>476</td>
<td>148</td>
<td>227</td>
<td>422</td>
<td>53</td>
</tr>
</tbody>
</table>

Table 6.1. True Positive (TP), True Negative (TN), False Negative (FN) reported by each tool variant on in-house evaluation criteria.

<table>
<thead>
<tr>
<th>Predictions/ Ground truth</th>
<th>O</th>
<th>AE</th>
<th>DATE</th>
<th>DRUG</th>
<th>NEGATION</th>
<th>PROBLEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>8850</td>
<td>113</td>
<td>18</td>
<td>15</td>
<td>36</td>
<td>21</td>
</tr>
<tr>
<td>AE</td>
<td>144</td>
<td>355</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>DATE</td>
<td>21</td>
<td>0</td>
<td>406</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DRUG</td>
<td>33</td>
<td>4</td>
<td>0</td>
<td>345</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>NEGATION</td>
<td>34</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>251</td>
<td>0</td>
</tr>
<tr>
<td>PROBLEM</td>
<td>48</td>
<td>47</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>79</td>
</tr>
</tbody>
</table>

Table 6.2. Confusion matrix by Stanza variant 1 on Dev set

<table>
<thead>
<tr>
<th>Predictions/ Ground truth</th>
<th>O</th>
<th>AE</th>
<th>DATE</th>
<th>DRUG</th>
<th>NEGATION</th>
<th>PROBLEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>9813</td>
<td>48</td>
<td>31</td>
<td>13</td>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td>AE</td>
<td>137</td>
<td>583</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>DATE</td>
<td>32</td>
<td>0</td>
<td>584</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DRUG</td>
<td>61</td>
<td>5</td>
<td>0</td>
<td>371</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>NEGATION</td>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>286</td>
<td>1</td>
</tr>
<tr>
<td>PROBLEM</td>
<td>38</td>
<td>25</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>113</td>
</tr>
</tbody>
</table>

Table 6.3. Confusion matrix by Stanza variant 1 on Test set