SymptoGraph: Evaluating BiLSTM vs. CNN for Symptom-Based Disease Prediction with Knowledge Graph Integration

With acknowledgment to:
Dr. Chen-Fu Chiang, PhD

Presented by:
Shireesh Reddy Pyreddy - U00345206

Introduction

- Recent years have seen significant progress in applying artificial intelligence to healthcare, particularly in disease prediction and diagnosis.

- Traditional machine learning techniques are effective but may struggle with the complexity of medical data. Deep learning, employing architectures like BiLSTM and CNN, offers enhanced capabilities in processing intricate datasets.

- Symptom-based disease prediction faces challenges due to symptom overlap and variations in presentation. Current models often lack the ability to understand the context and semantics of symptom data, creating a need for more sophisticated approaches.

- Introducing SymptoGraph, a novel framework that combines the strengths of BiLSTM and CNN with knowledge graphs. By integrating knowledge graphs, the model aims to enhance context-aware predictions, providing a structured representation of relationships and semantics between medical entities.

Motivation

- The convergence of Natural Language Processing (NLP) and healthcare presents a fertile ground for transformative advancements, offering opportunities to revolutionize disease prediction and patient care.
- Leveraging NLP methodologies such as Text Classification, Feature Extraction, Named Entity Recognition, and Knowledge Graph Generation to extract meaningful insights from unstructured patient narratives and decipher intricate patterns in symptom-based disease prediction.
- Focusing on the comparison of two powerful neural network architectures—Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Network (CNN)—to understand their effectiveness in unraveling latent structures within symptom narratives.
- Introducing a novel dimension by integrating a knowledge graph to capture contextual relationships in healthcare data, aiming to provide a structured foundation for predictive models and enhance interpretability.

Related Works

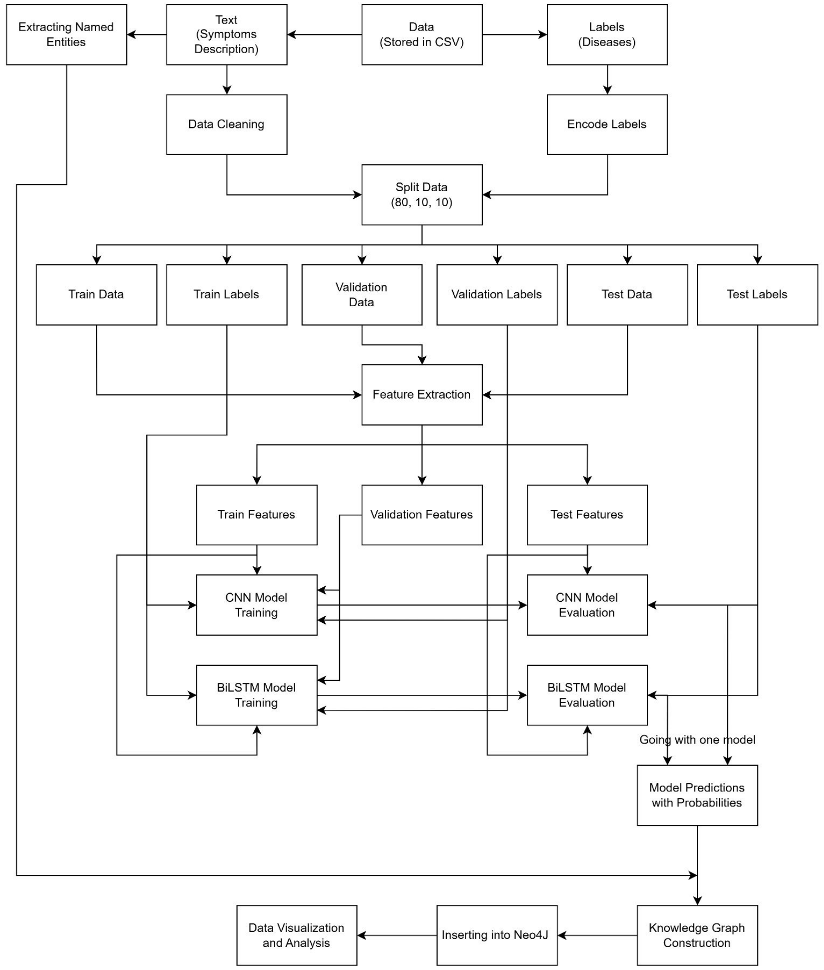
- Disease Prediction using Symptoms based on Machine Learning Algorithms.
- Identification and Prediction of Chronic Diseases Using Machine Learning Approach.
- Disease Prediction From Various Symptoms Using Machine Learning.
- Disease Prediction Based on Symptoms Given by User Using Machine Learning.

What’s New - The Uncommon Factor

- **Previous Approaches:**
  - In the most of the previous works, the problem was addressed using machine learning approaches like Random Forest, KNN, SVM etc.

- **Our Approach:**
  - Our approach will include training the two deep learning models (BiLSTM and CNN) from scratch and comparing both of them.
  - Also, our approach will extend it further by creating a knowledge graph to understand the semantics and relations of the symptoms and diseases which no one did it before.

The Architecture - Overview



The Dataset

Statistics

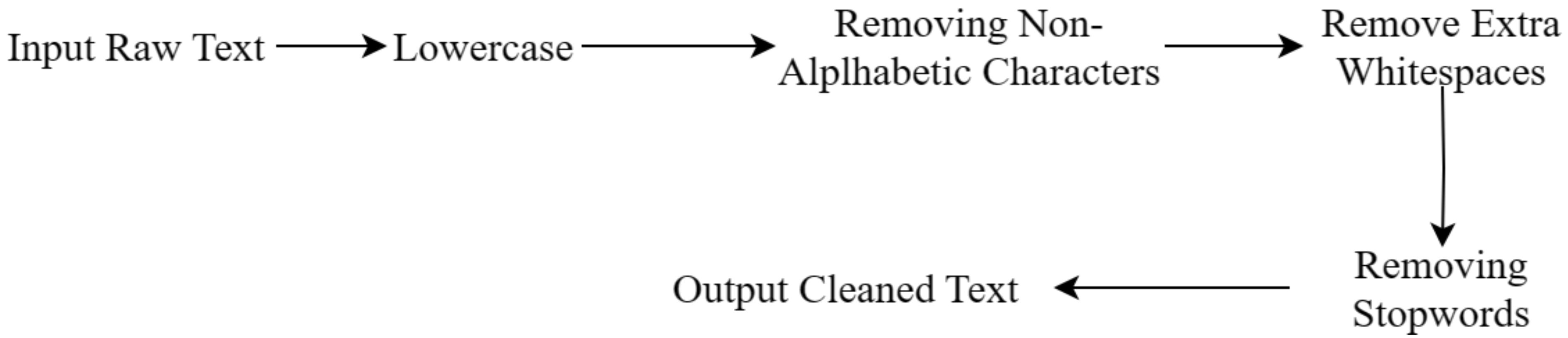
- **24** different diseases.
- Each disease has **50** symptom descriptions.
- Total **1200** data points.
- **The following 24 diseases have been covered in the dataset:**
  Psoriasis, Varicose Veins, Typhoid, Chicken pox, Impetigo, Dengue, Fungal infection, Common Cold, Pneumonia, Dimorphic Hemorrhoids, Arthritis, Acne, Bronchial Asthma, Hypertension, Migraine, Cervical spondylosis, Jaundice, Malaria, urinary tract infection, allergy, gastroesophageal reflux disease, drug reaction, peptic ulcer disease, diabetes

Complexities

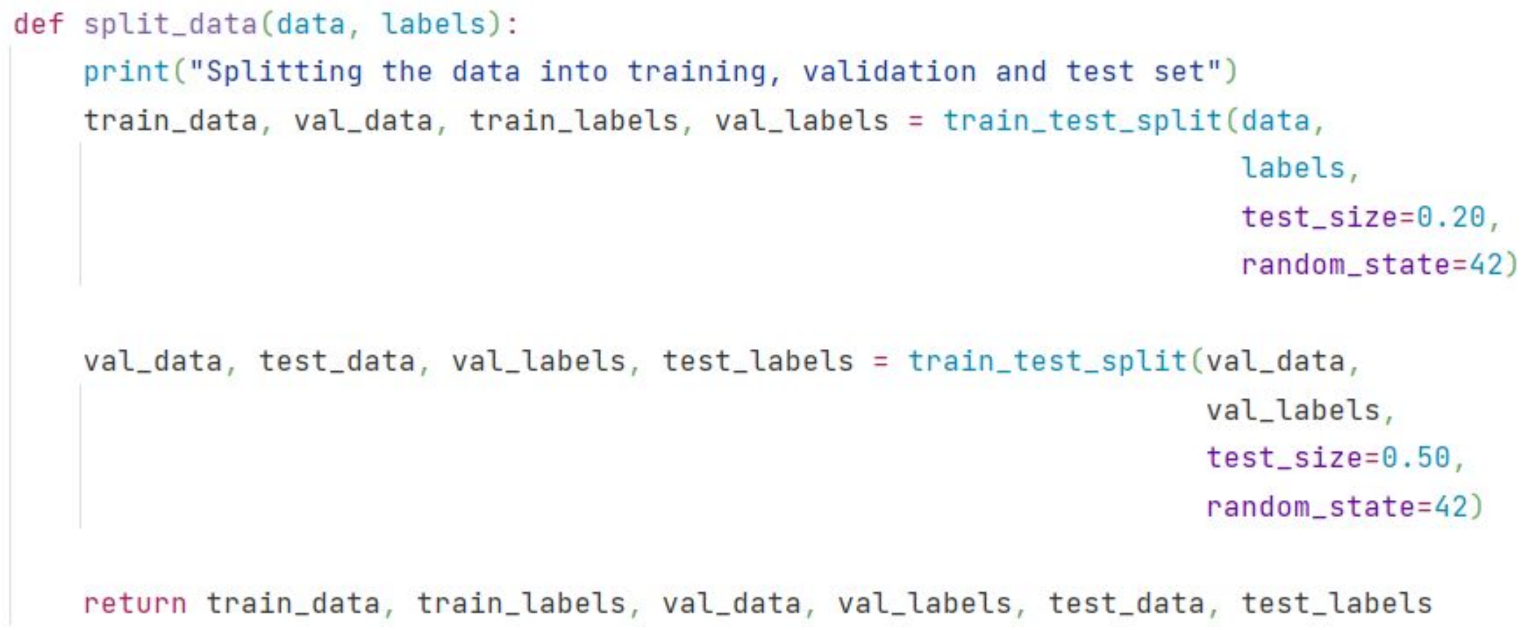
- Classifying 24 classes is hard and challenging for a text classification problem.
- Ambiguity across different diseases.
- Less data to train deep learning models like BiLSTM and CNN.
- Training Named Entity Recognition is challenging considering the ambiguity across different entities and the computational resources.

Data Preprocessing

Data Cleaning - Pipeline

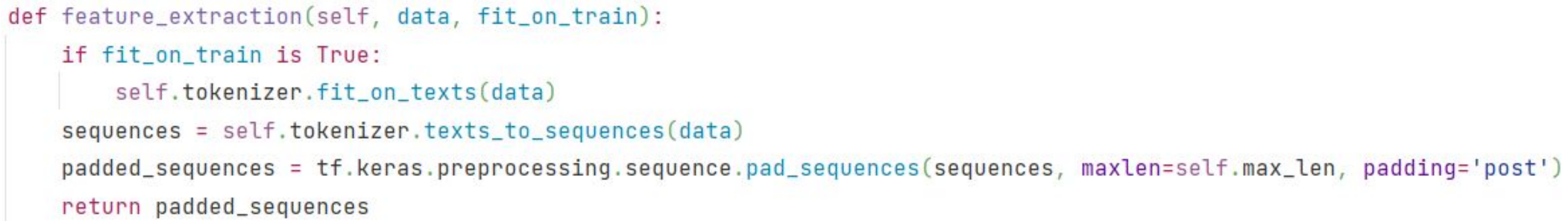


Data Split



```python
def split_data(data, labels):
    print("Splitting the data into training, validation and test set")
    train_data, val_data, train_labels, val_labels = train_test_split(data,
                                                                      labels,
                                                                      test_size=0.20,
                                                                      random_state=42)

    val_data, test_data, val_labels, test_labels = train_test_split(val_data,
                                                                    val_labels,
                                                                    test_size=0.50,
                                                                    random_state=42)

    return train_data, train_labels, val_data, val_labels, test_data, test_labels
```

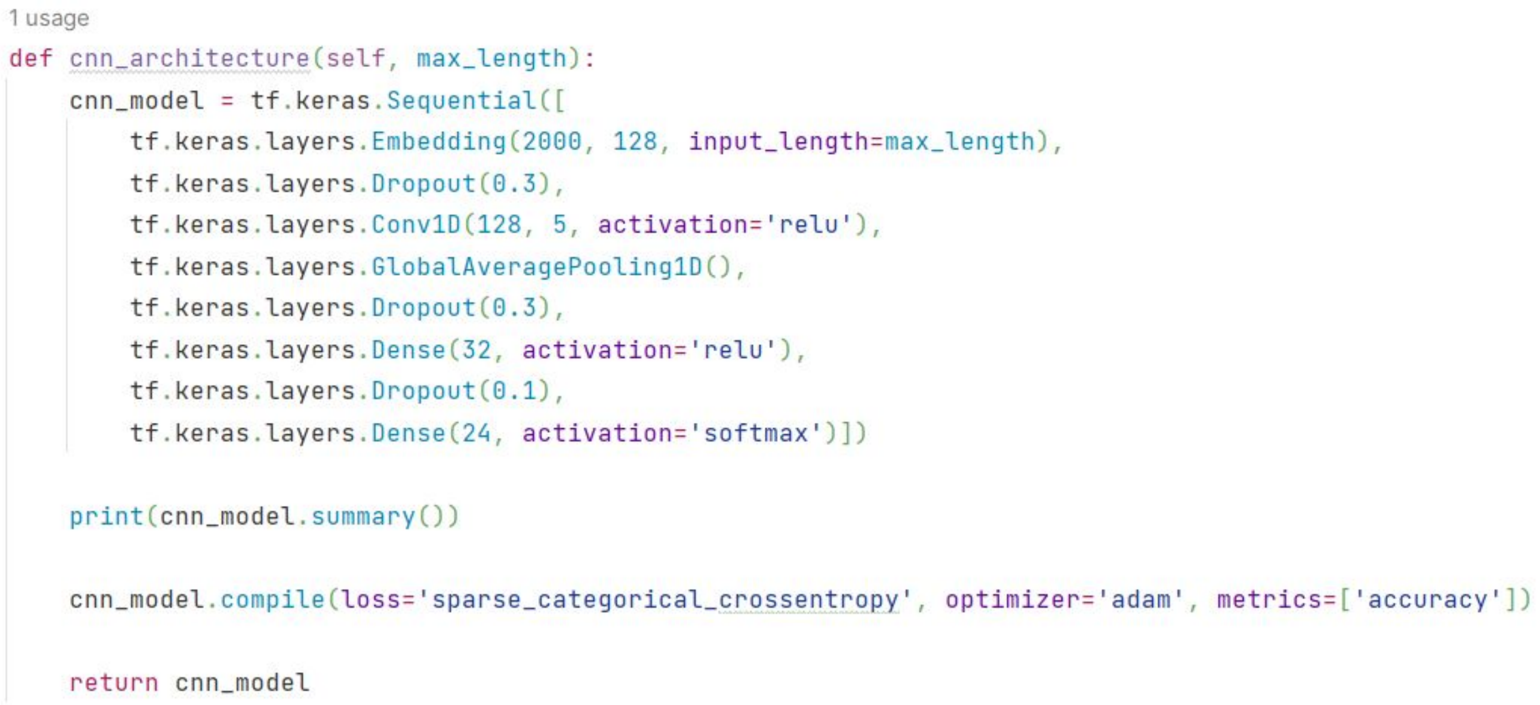
Data Transformation



```python
def feature_extraction(self, data, fit_on_train):
    if fit_on_train is True:
        self.tokenizer.fit_on_texts(data)
    sequences = self.tokenizer.texts_to_sequences(data)
    padded_sequences = tf.keras.preprocessing.sequence.pad_sequences(sequences, maxlen=self.max_len, padding='post')
    return padded_sequences
```

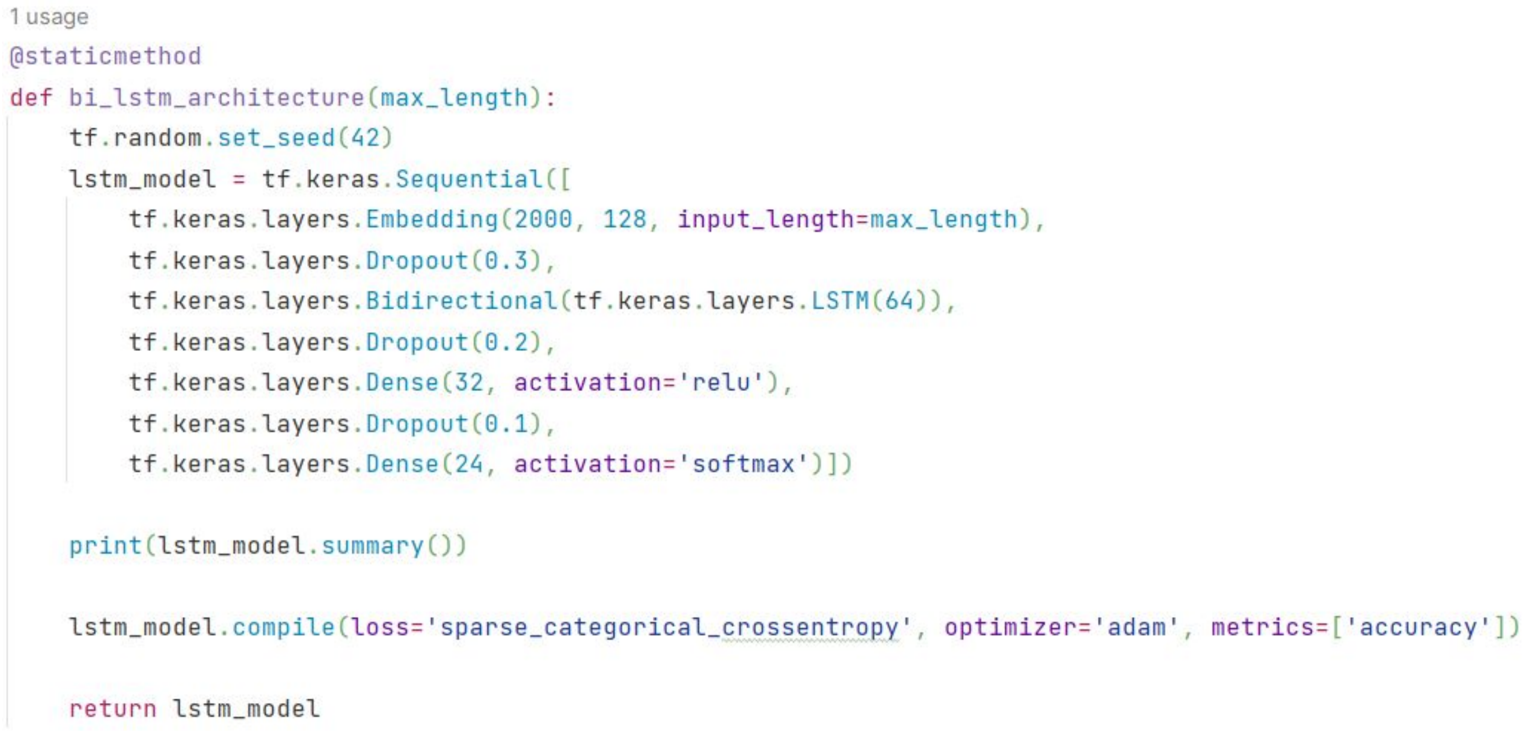
Model Designing

CNN Architecture Designing



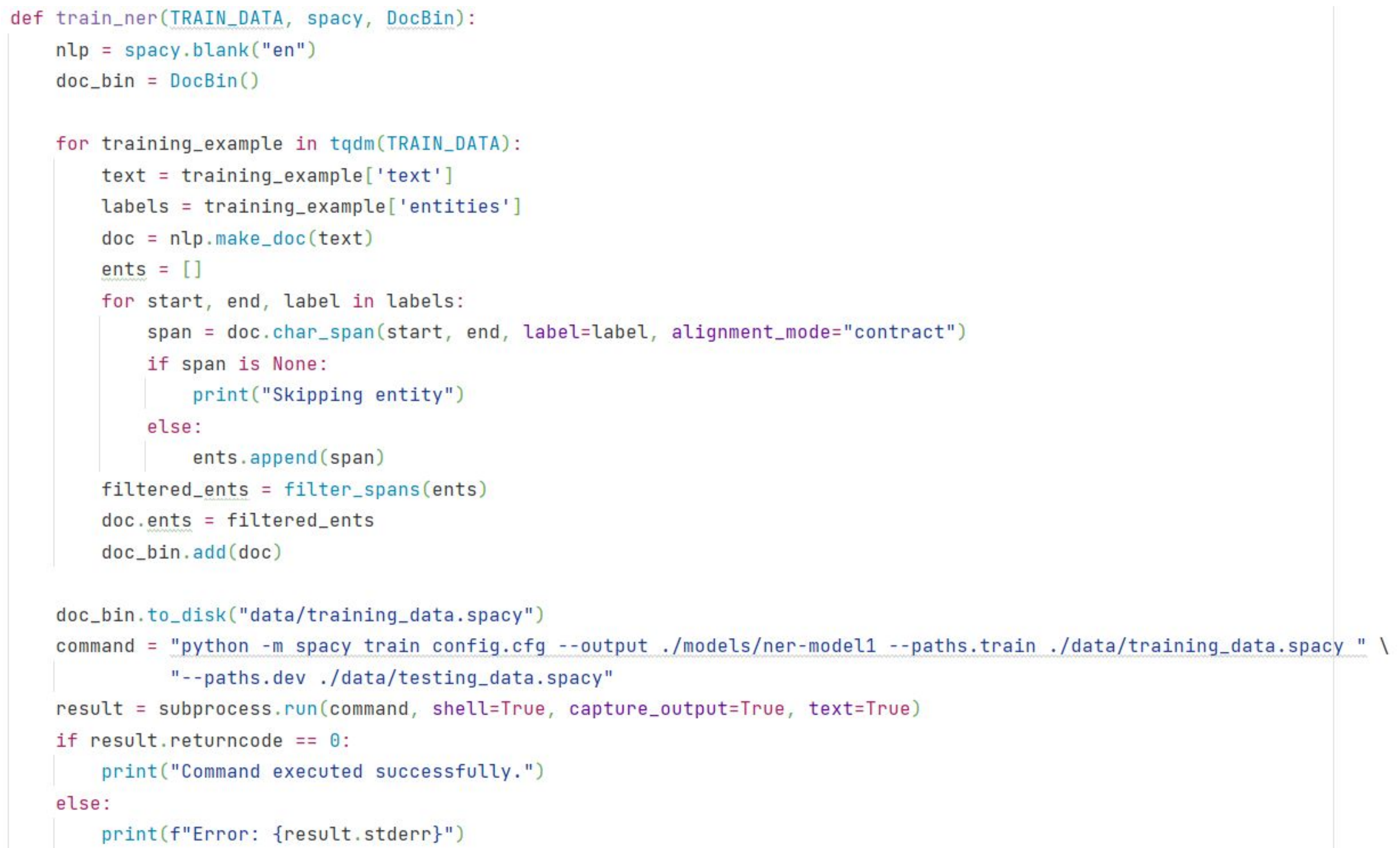
```
1 usage
def cnn_architecture(self, max_length):
    cnn_model = tf.keras.Sequential([
        tf.keras.layers.Embedding(2000, 128, input_length=max_length),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Conv1D(128, 5, activation='relu'),
        tf.keras.layers.GlobalAveragePooling1D(),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dropout(0.1),
        tf.keras.layers.Dense(24, activation='softmax')])

    print(cnn_model.summary())

    cnn_model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    return cnn_model
```

BiLSTM Architecture Designing



```
1 usage
@staticmethod
def bi_lstm_architecture(max_length):
    tf.random.set_seed(42)
    lstm_model = tf.keras.Sequential([
        tf.keras.layers.Embedding(2000, 128, input_length=max_length),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dropout(0.1),
        tf.keras.layers.Dense(24, activation='softmax')])

    print(lstm_model.summary())

    lstm_model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    return lstm_model
```

Named Entity Recognition Architecture Designing



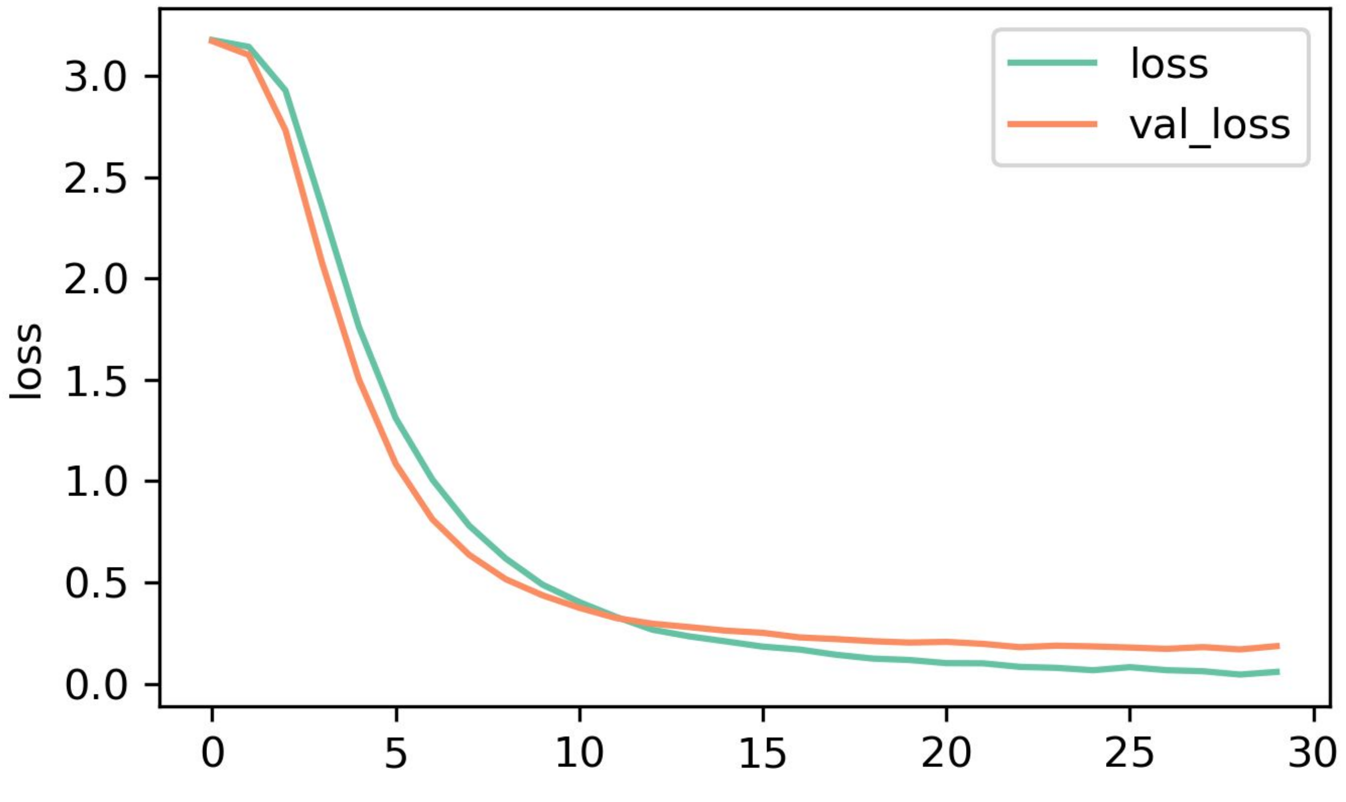
```python
def train_ner(TRAIN_DATA, spacy, DocBin):
    nlp = spacy.blank("en")
    doc_bin = DocBin()

    for training_example in tqdm(TRAIN_DATA):
        text = training_example['text']
        labels = training_example['entities']
        doc = nlp.make_doc(text)
        ents = []
        for start, end, label in labels:
            span = doc.char_span(start, end, label=label, alignment_mode="contract")
            if span is None:
                print("Skipping entity")
            else:
                ents.append(span)
        filtered_ents = filter_spans(ents)
        doc.ents = filtered_ents
        doc_bin.add(doc)

    doc_bin.to_disk("data/training_data.spacy")
    command = "python -m spacy train config.cfg --output ./models/ner-model1 --paths.train ./data/training_data.spacy " \
              "--paths.dev ./data/testing_data.spacy"
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    if result.returncode == 0:
        print("Command executed successfully.")
    else:
        print(f"Error: {result.stderr}")
```

Named Entity Recognition - Hyperparameters

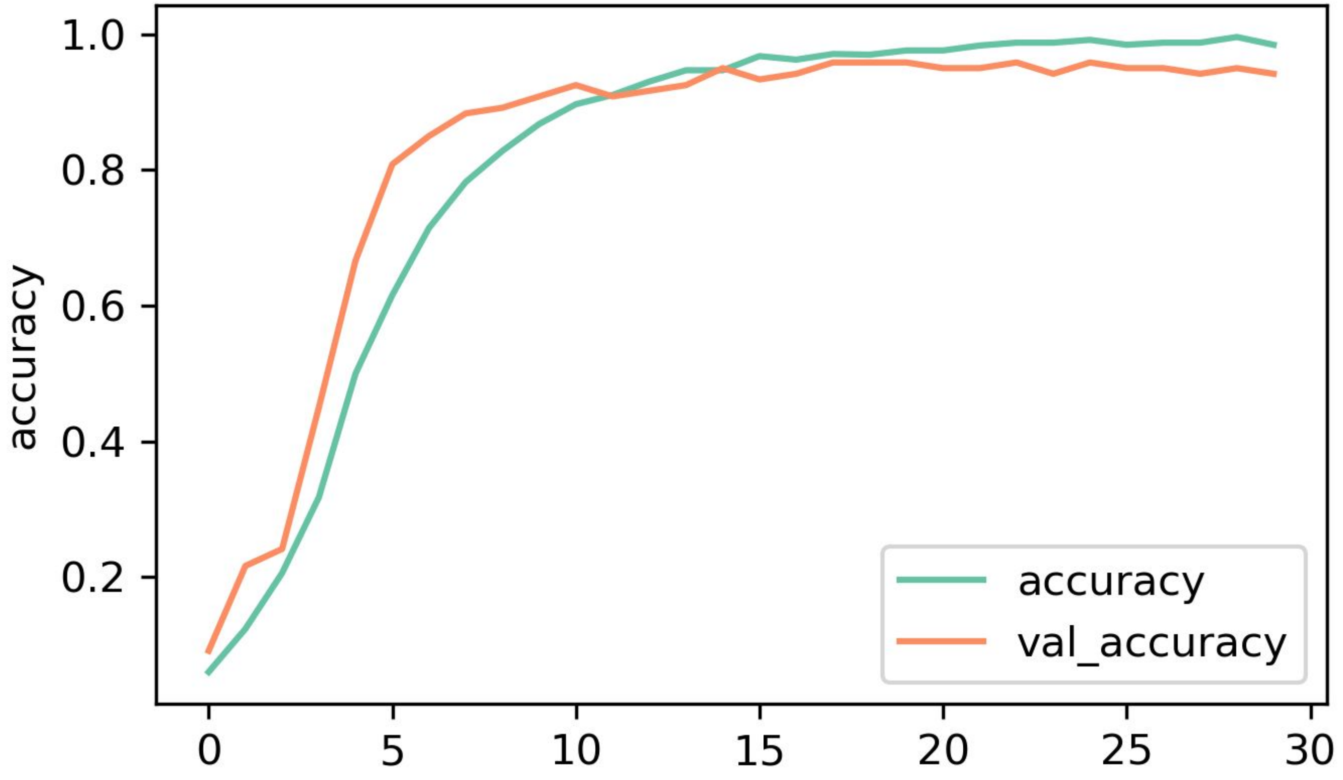
- `lang = "en"`
- `batch_size = 1000`
- `tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}`
- `dropout = 0.1`
- `epochs = 57`
- `@optimizers = "Adam.v1"`
- `beta1 = 0.9`
- `beta2 = 0.999`
- `L2_is_weight_decay = true`
- `L2 = 0.01`
- `grad_clip = 1.0`
- `eps = 0.00000001`
- `learn_rate = 0.001`

Model Training and Evaluation

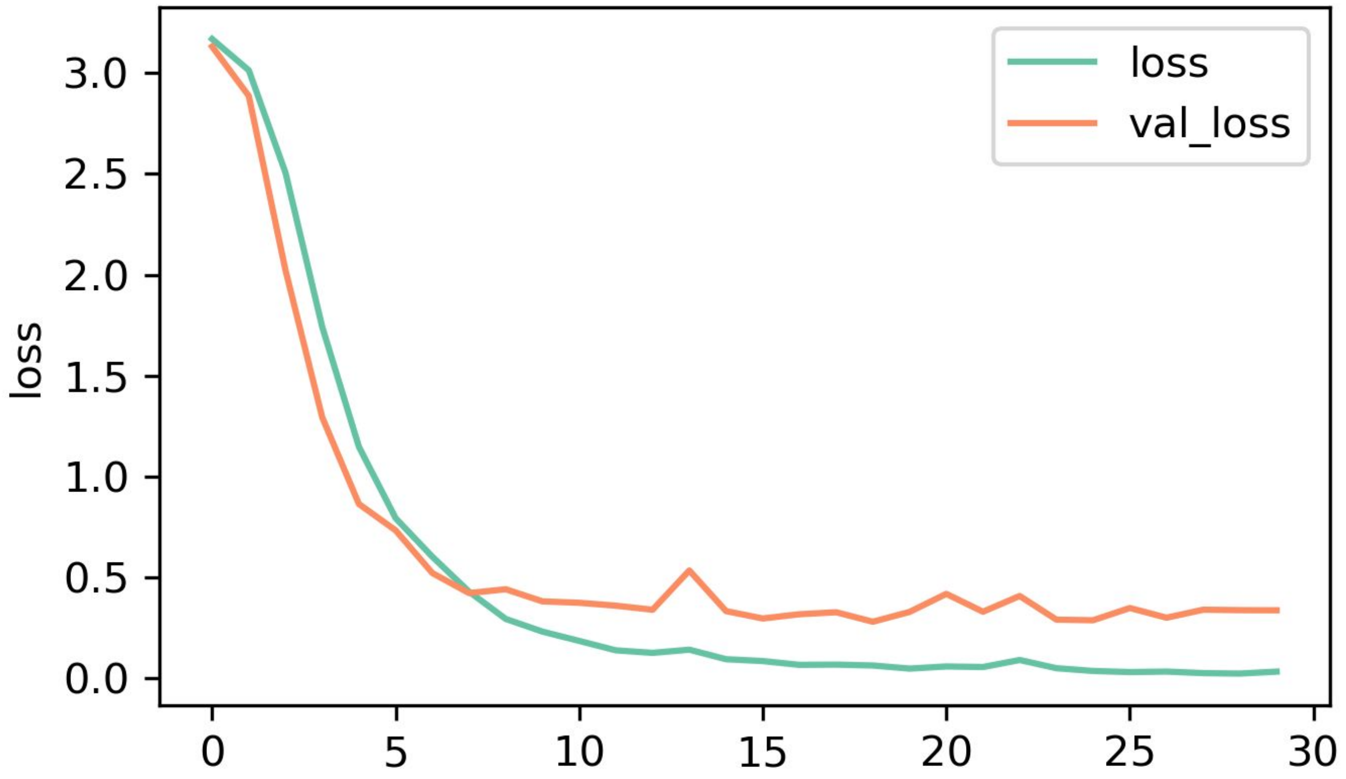
CNN Training and Validation Loss



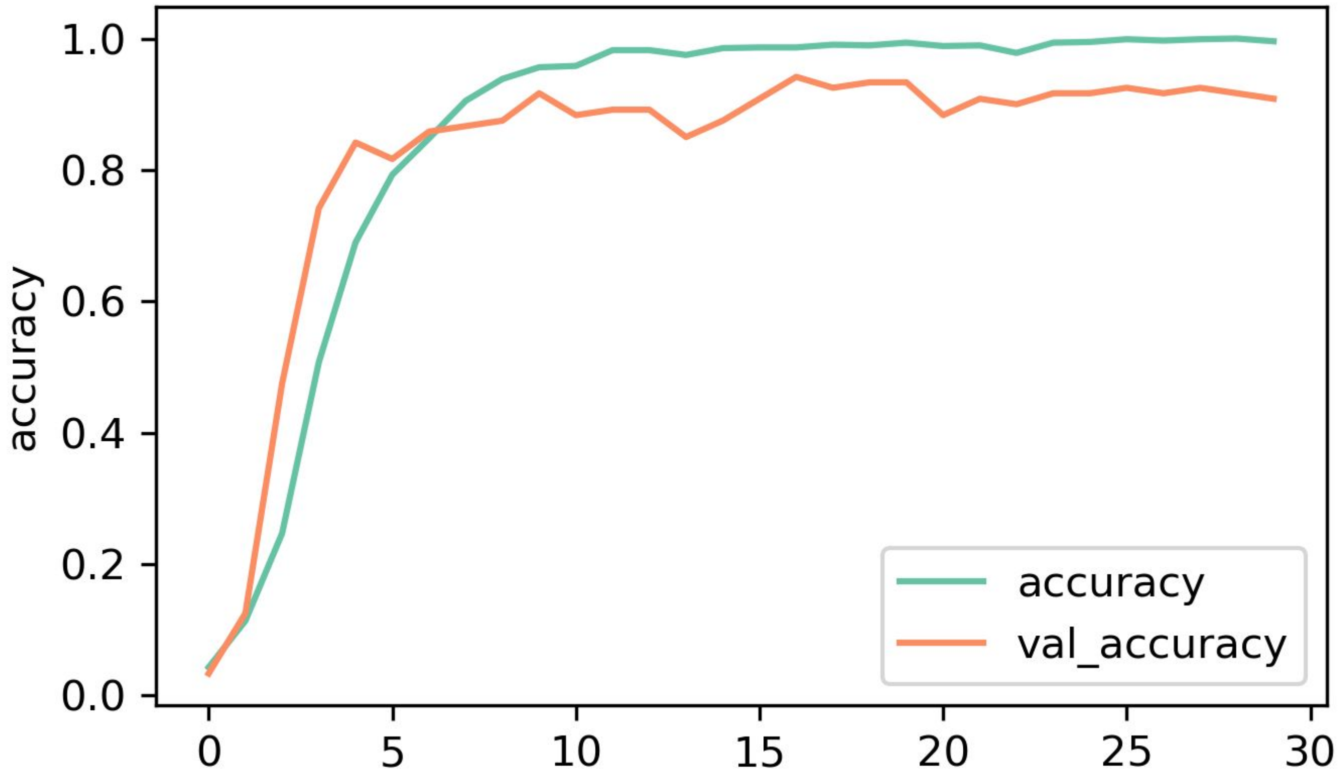
CNN Training and Validation Accuracy



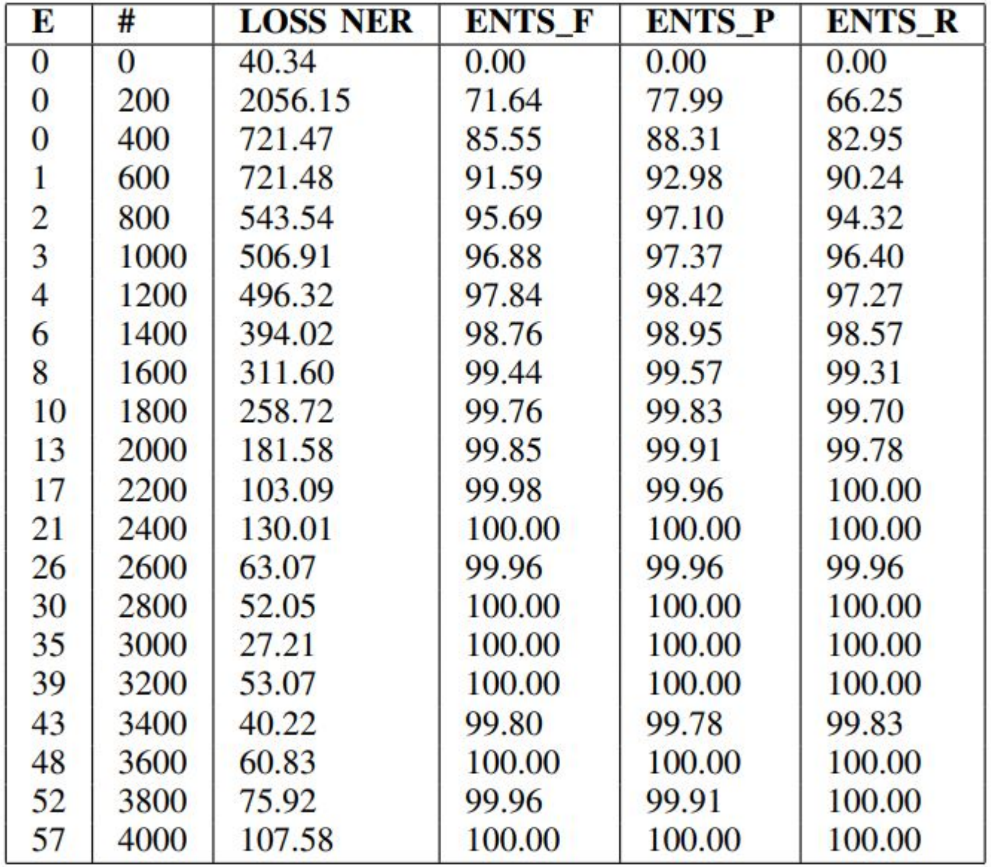
BiLSTM Training and Validation Loss



BiLSTM Training and Validation Accuracy



NER Training and Validation Details



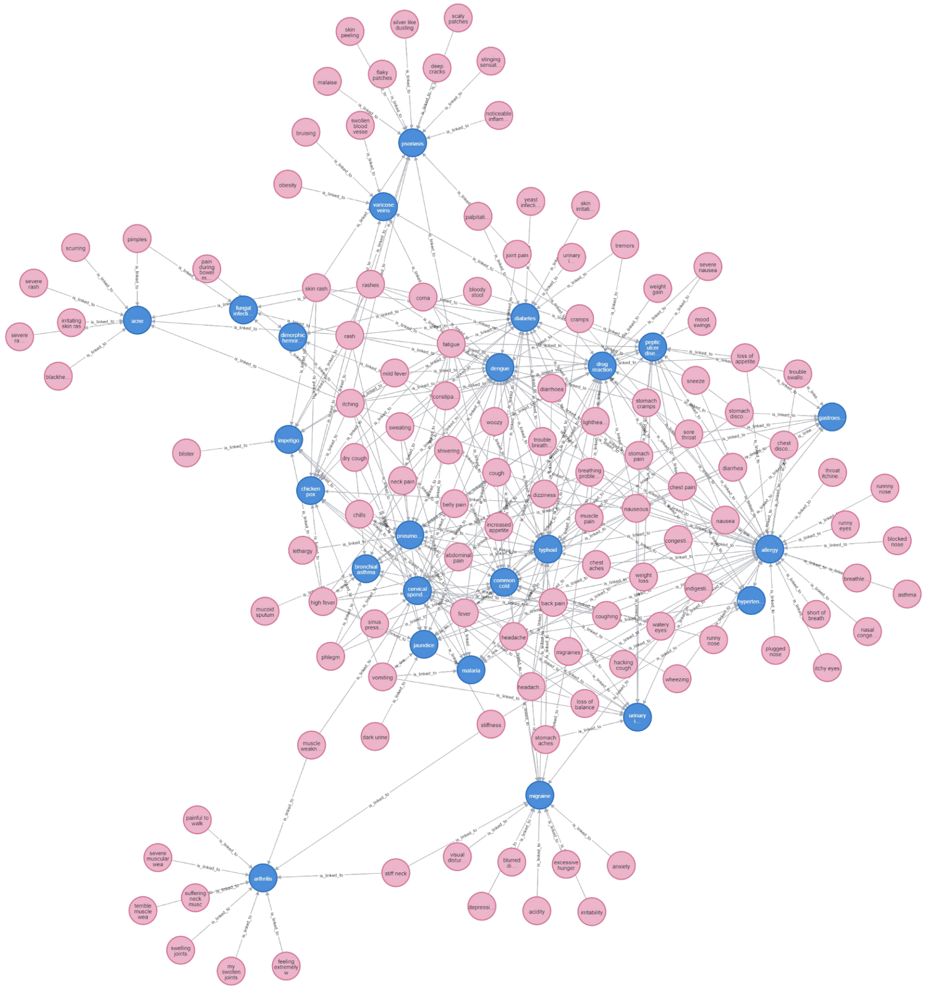
E	#	LOSS NER	ENTS_F	ENTS_P	ENTS_R
0	0	40.34	0.00	0.00	0.00
0	200	2056.15	71.64	77.99	66.25
0	400	721.47	85.55	88.31	82.95
1	600	721.48	91.59	92.98	90.24
2	800	543.54	95.69	97.10	94.32
3	1000	506.91	96.88	97.37	96.40
4	1200	496.32	97.84	98.42	97.27
6	1400	394.02	98.76	98.95	98.57
8	1600	311.60	99.44	99.57	99.31
10	1800	258.72	99.76	99.83	99.70
13	2000	181.58	99.85	99.91	99.78
17	2200	103.09	99.98	99.96	100.00
21	2400	130.01	100.00	100.00	100.00
26	2600	63.07	99.96	99.96	99.96
30	2800	52.05	100.00	100.00	100.00
35	3000	27.21	100.00	100.00	100.00
39	3200	53.07	100.00	100.00	100.00
43	3400	40.22	99.80	99.78	99.83
48	3600	60.83	100.00	100.00	100.00
52	3800	75.92	99.96	99.91	100.00
57	4000	107.58	100.00	100.00	100.00

Results

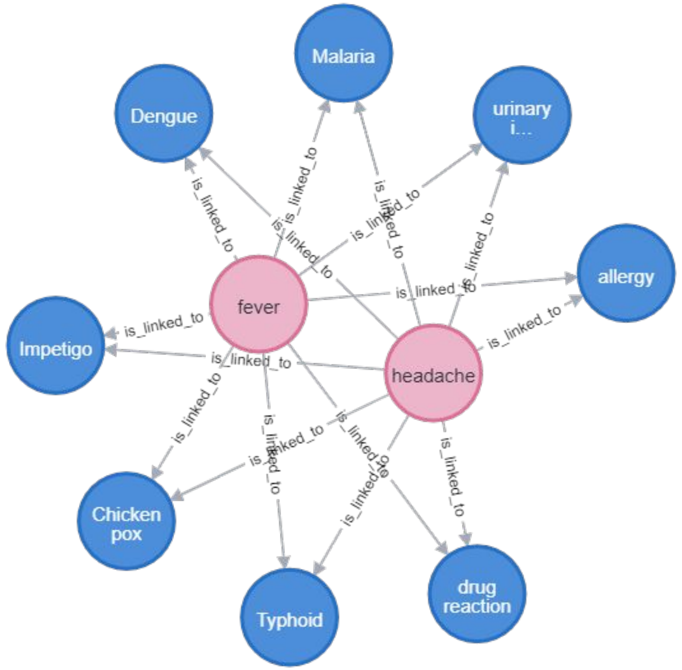
CNN vs LSTM

Model	Accuracy	Precision	Recall	F1-Score
CNN	0.91	0.92	0.90	0.90
BiLSTM	0.88	0.84	0.84	0.83

The Knowledge Graph - The High Level View

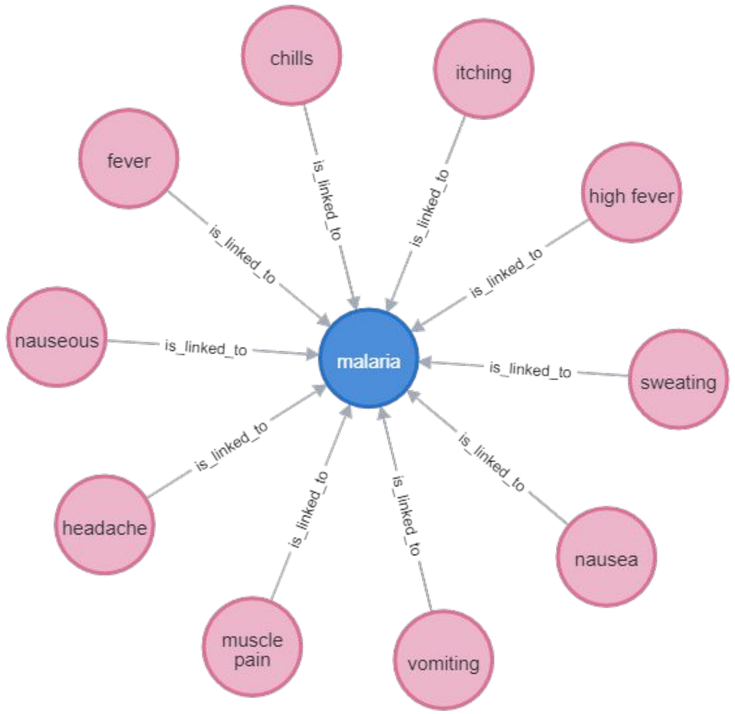


Analysis Using Knowledge Graph



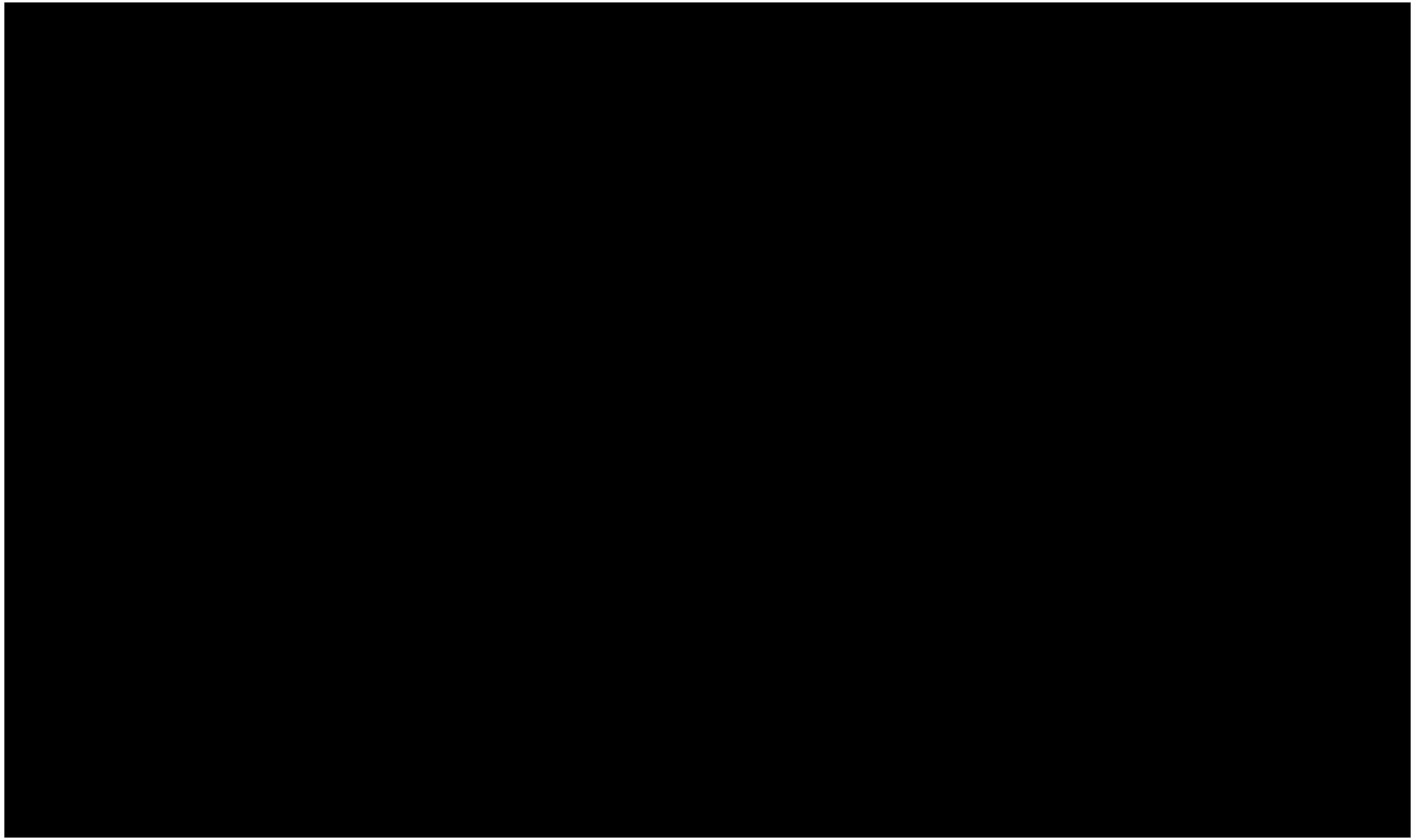
For example when a patient presents early stage symptoms like "fever" and "headache", a doctor can refer to this graph to explore potential diseases that are associated with these symptoms.

Analysis Using Knowledge Graph (Part 2)



Extending the previous example, by examining the other symptoms linked to each potential disease in the graph like this, the doctor can cross check with the patient's symptoms. This could help narrow down the possible diseases and guide further diagnostic tests or treatments.

Demo



Conclusion

- SymptoGraph represents a notable advancement in symptom-based disease prediction, utilizing advanced deep learning models and integrating knowledge graphs at the intersection of Natural Language Processing (NLP) and healthcare.

- Comparative analysis of Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Network (CNN) models, along with the incorporation of a knowledge graph, contributes to enhanced interpretability in SymptoGraph.

- Important to acknowledge challenges, including dataset size, ambiguity in symptom descriptions, and ethical considerations, emphasizing the need for addressing these limitations to refine SymptoGraph's robustness.

- SymptoGraph, despite current constraints, lays a foundation for future innovations in leveraging NLP for nuanced disease prediction. Opportunities include dataset augmentation, improved entity recognition, and a deeper exploration of ethical implications, contributing to more informed healthcare decision-making.

Thanks for listening.

Any Questions?