# Software Design Documentation for

## Task Extraction

## and

## Management System

Prepared by:
Abineet Thampi

# Introduction

This project aims to develop an automated system for extracting tasks, deadlines, and recipients from natural language text inputs using Named Entity Recognition (NER). The system is designed to help users efficiently organize and track their tasks and deadlines, improving productivity and time management.

# Study of Existing Implementations

Existing NER systems often struggle with domain-specific tasks like extracting task-related information. This implementation seeks to improve upon existing solutions by using a custom-trained NER model based on BERT to accurately extract task-related information from various text inputs.

# AI Technologies and Models Used

**The project utilizes the following AI technologies and models:**

- Named Entity Recognition (NER): A custom NER model is trained to identify and extract tasks, deadlines, and recipients from text.
- Natural Language Processing (NLP): Used for processing and understanding text inputs.
- DistilBERT: A pre-trained transformer model used as the foundation for creating and fine-tuning the custom NER model.
- Hugging Face Transformers: A library providing state-of-the-art NLP models and tools.
- Date parsing: The dateutil library is used to standardize and process various date formats.

# Description of the Dataset

The dataset consists of labeled examples of task-related sentences and paragraphs. Each example contains annotations for tasks, deadlines, and recipients. The dataset covers a wide range of task types, deadline formats, and recipient mentions, ensuring that the model can handle diverse inputs.

# Dependencies

**Software Dependencies:**

- Python 3.7+
- transformers
- torch
- datasets
- sklearn
- numpy
- seqeval
- sklearn
- nltk

**Hardware Dependencies:**

- CPU: Multi-core processor (4+ cores recommended for faster training)
- RAM: 8GB minimum, 16GB or more recommended
- Storage: At least 1GB of free disk space for model storage and data

**Optional Hardware for Improved Performance:**

- GPU: NVIDIA GPU with CUDA support for faster model training

# Detailed Workflow

## dbmodel.py

Key Functions:

- augment_data(examples, num_augmented_examples=2): Performs data augmentation using entity replacement.
- entity_replacement(text, annotations): Replaces entities in the text with randomly chosen entities of the same type.
- preprocess_function(examples): Tokenizes and aligns labels for the NER task.
- compute_metrics(p): Computes evaluation metrics using the seqeval framework.
- train_model(): Sets up and trains the NER model using the Hugging Face Trainer.

## dbapp.py

Key Functions:

- extract_tasks_and_deadlines(text, model, tokenizer): Uses the trained NER model to extract tasks, deadlines, and recipients from the input text.
- check_and_create_file(file_path): Checks if the Excel file exists, creates it if it doesn't.
- write_to_excel(file_path, tasks, deadlines): Writes the extracted tasks and deadlines to the Excel file.

# Environment Setup

- Install required Python libraries: transformers, torch, datasets, numpy
- Set up a virtual environment (recommended)

# Data Preparation

- Prepare labeled data in the format: (text, {"entities": [(start, end, label), ...]})
- Implement data augmentation techniques (entity replacement)
- Ensure diverse examples covering various task types, deadline formats, and recipient mentions

# Model Training (dbmodel.py)

- Load and preprocess training data
- Initialize a pre-trained BERT model for token classification
- Define training parameters (learning rate, batch size, number of epochs)
- Train the model using the Hugging Face Trainer
- Implement evaluation using seqeval metrics
- Save the trained model to disk

# Task Extraction and Management (dbapp.py)

- Model Loading:
  - Load the trained NER model from disk
- User Interface:
  - Implement a command-line interface for user input
- Entity Extraction:
  - Process each input text using the loaded NER model
  - Extract tasks, deadlines, and recipients
- Output:
  - Display extracted information to the user

# System Workflow Diagram

**[User Input] -> [NER Model] -> [Entity Extraction]**
**|**
**v**
**[Display Results]**

# Usage Instructions

- Ensure all dependencies are installed and the environment is set up.
- Run 'dbmodel.py' to train the NER model (if not already trained).
- Execute 'dbapp.py' to start the task extraction system.
- Enter task-related text when prompted.
- Review the extracted tasks, deadlines, and recipients displayed in the console.

**Step-by-Step Process:**

1) Data Preparation:
- Collect and annotate task-related text samples.
- Format the data as a list of tuples, each containing the text and entity annotations.
- Define label mappings for entity types (TASK, DEADLINE, RECIPIENT).

2) Model Setup:

- Import necessary libraries (transformers, datasets, evaluate).
- Load the DistilBERT tokenizer and model for token classification.
- Define data collator for token classification tasks.

3) Data Preprocessing:

- Create a function to tokenize and align labels with input tokens
- Apply the preprocessing function to the entire dataset using the map function.
- Split the dataset into training and validation sets

4) Model Training:

- Set up training arguments (learning rate, batch size, number of epochs)
- Initialize the Trainer with the model, training arguments, datasets, and evaluation metric
- Implement data augmentation techniques (entity swapping, entity replacement)
- Train the model using the Trainer's train() method

5) Evaluation:

- Define a compute_metrics function to calculate precision, recall, and F1 score
- Use the Trainer's evaluate() method to assess model performance on the validation set

6) Inference:

- Load the trained model and tokenizer
- Create a pipeline for token classification using the trained model
- Process input text through the pipeline to extract entities (tasks, deadlines, recipients)

7) Post-processing:

- Implement functions to standardize extracted information (e.g., date formats for deadlines)
- Group extracted entities into coherent task entries

8) Data Storage:

- Set up a mechanism to store extracted information (e.g., Excel file or database)
- Implement functions to write extracted tasks, deadlines, and recipients to the storage medium

9) User Interface:

- Create a simple command-line interface for user input
- Allow users to input text and display extracted task information.

# Training Metrics

The model's training progress was monitored using the Hugging Face Trainer's built-in logging and evaluation features.

Here is the training configuration:

- Learning rate: 2e-5
- Batch size: 16 and 32(for training and evaluation)
- Number of epochs: 6
- Weight decay: 0.01
- Evaluation strategy: Evaluated every few steps
- Best model selection: Based on the highest F1 score

# Evaluation Metrics

The model's performance is evaluated using the seqeval framework and the trainer’s built-in feature, which provides the following metrics:

- Precision
- Recall
- F1 score

**Overall Performance:**

- Precision: 0.6376 (63.76%)
- Recall: 0.7138 (71.38%)
- F1 Score: 0.6736 (67.36%)

**Performance by Entity Type:**

a. DEADLINE:

- Precision: 0.5475 (54.75%)
- Recall: 0.5731 (57.31%)
- F1 Score: 0.5600 (56.00%)

b. RECIPIENT:

- Precision: 0.6797 (67.97%)
- Recall: 0.7761 (77.61%)
- F1 Score: 0.7247 (72.47%)

c. TASK:

- Precision: 0.6667 (66.67%)
- Recall: 0.7682 (76.82%)
- F1 Score: 0.7138 (71.38%)

# Findings

- The performance of the model can be further improved by:
  - Increasing the quality and quantity of the training dataset
  - Fine-tuning hyper parameters
- The loss has decreased, indicating better model fit
- Performance improved for all entity types
- The model also performs better in identifying tasks from inputs with multiple sentences

# References


- Hugging Face. (n.d.). Transformers. https://huggingface.co/transformers/
- Medium.(2024) Fine-tune Your Own BERT Token Classification Model. https://medium.com/@raj.pulapakura/fine-tune-your-own-bert-token-classification-model-06b1153fbf56.