MedTalk-Dz Voice-Based Disease Prediction System Using Speech-to-Symptoms

Mini Projet Natural Language Processing

Submitted by:

- *Larbi Mohammed Achraf*
- *Ouddane Youcef Fahd*
- *Ourred Islem Charaf Eddine*
- *Benbekarti Mohammed El Amine*

Supervised by:

Prof. Rabab BOUSMAHA

Contents

Abstract

MEDTalk-Dz is a pioneering system designed to facilitate preliminary medical diagnosis by converting spoken Algerian dialect into symptom data and predicting diseases using a neural network. The project comprises two main components: (1) Speech-to-Symptoms, which records voice input, transcribes it using the ElevenLabs API, translates and extracts symptoms using OpenAI's GPT-3.5, and maps unmatched symptoms to the dataset using Sentence Transformers; and (2) Disease Prediction, which employs a Multi-Layer Perceptron (MLP) neural network trained on the SymbiPredict 2022 dataset. This report provides an in-depth exploration of the methodologies, including data preprocessing, model training, hyperparameter tuning, and symptom mapping with Sentence Transformers. It also presents code implementations, results with visual outputs, and discusses the system's potential in healthcare diagnostics.

1 Problem Overview

Medical diagnosis often depends on accurately understanding symptoms described by patients. In regions like Algeria, where individuals commonly express themselves in dialectal Arabic rather than formal medical language, this poses a significant challenge for automated systems. The linguistic gap between user-described symptoms and structured medical datasets makes it difficult to process and interpret inputs effectively.

Furthermore, most AI-based diagnostic tools require users to select symptoms from predefined lists, which may not be intuitive or accessible for non-expert users. This limitation can result in poor user experience, misdiagnosis, or a lack of engagement with digital health platforms.

The MEDTalk-Dz system addresses these issues by enabling users to describe symptoms naturally in the Algerian dialect. The system captures voice input, transcribes and translates it, extracts relevant symptoms, maps them to a standardized feature space, and uses deep learning techniques to predict possible diseases. This approach helps make diagnostic support more accessible, especially in under-resourced or linguistically diverse settings.

2 Introduction

MEDTalk-Dz addresses the challenge of accessible medical diagnostics by enabling users to describe symptoms in the Algerian dialect and receive potential disease predictions. By integrating natural language processing (NLP), speech recognition, and deep learning, the system offers a novel approach to preliminary diagnosis. The project is divided into two core components:

- **Speech-to-Symptoms**: Captures and processes voice input to extract symptoms, using ElevenLabs for transcription, GPT-3.5 for translation and extraction, and Sentence Transformers for symptom mapping.
- **Disease Prediction**: Utilizes a Multi-Layer Perceptron (MLP) neural network to predict diseases based on extracted symptoms.

This report expands on the methodologies, providing detailed workflows, code implementations, and visual results to demonstrate the system's efficacy. It also introduces the use of Sentence Transformers to enhance symptom mapping, ensuring robustness when extracted symptoms do not directly match the dataset's vocabulary.

3 Project Workflow

The MEDTalk-Dz system follows a comprehensive workflow:

1. **Voice Recording**: Captures user speech in the Algerian dialect.
2. **Speech Transcription**: Converts audio to text using ElevenLabs API.
3. **Translation**: Translates transcribed text to English using GPT-3.5.
4. **Symptom Extraction**: Identifies medical symptoms from translated text.
5. **Symptom Mapping**: Uses Sentence Transformers to map extracted symptoms to the datasets symptom vocabulary if direct matches are unavailable.
6. **Disease Prediction**: Feeds mapped symptoms into an MLP neural network to predict diseases.

4 Methodology

This section provides an exhaustive overview of the methodologies employed in MEDTalk-Dz, covering data collection, preprocessing, model design, training, and evaluation.

4.1 Data Collection

The project leverages two primary data sources:

- **Audio Input**: User-provided speech in the Algerian dialect, recorded as WAV or MP3 files.
- **SymbiPredict 2022 Dataset**: A structured dataset from Mendeley Data, containing 4,961 patient cases with 132 binary symptom features and 41 disease labels.

Audio data is collected in real-time during system operation, while the SymbiPredict dataset is preprocessed offline to train the MLP model.

4.2 Speech-to-Symptoms Methodology

The Speech-to-Symptoms component involves multiple stages, each with specific techniques to ensure accurate symptom extraction.

4.2.1 VNAN Voice Recording

Voice recording is performed using the `sounddevice` library, capturing 10 seconds of mono-channel audio at a 44.1 kHz sample rate. The audio is saved as a WAV file to ensure compatibility with downstream APIs.

This stage is a fundamental part of the VNAN (Voice-to-NLP-AI Network) pipeline, serving as the raw input capture for subsequent transcription and symptom processing. The parameters used in recordingsuch as duration, sample rate, and number of channelscan be easily modified to adapt to different environments or deployment constraints. For instance, in noisy settings, longer durations or noise suppression filters may be applied to enhance speech clarity. Similarly, the sample rate can be reduced to 16 kHz to optimize file size and transmission over bandwidth-limited connections without significant loss in transcription quality.

The modular nature of VNAN allows this audio acquisition component to be integrated or replaced independently, supporting future extensions such as real-time streaming or integration with embedded systems (e.g., Raspberry Pi). Moreover, the recording process can be looped or adjusted to perform repeated captures, silence detection, or automatic voice activity triggering, making the pipeline adaptable for both controlled and uncontrolled environments.

This flexibility ensures that the VNAN layer can be tuned to different field requirements, whether deployed in clinical settings, remote health kiosks, or mobile diagnostic units.

4.2.2 Speech Transcription with ElevenLabs

ElevenLabs provides advanced speech recognition services through its API, which can accurately transcribe audio into text, including the Algerian Arabic dialect. Their platform utilizes state-of-the-art models to process audio files in various languages, with a special emphasis on multilingual support, making it ideal for handling regional dialects like Algerian Arabic.

Platform Overview: ElevenLabs is a leading provider in the field of AI-driven speech transcription, leveraging deep learning models that are fine-tuned to recognize nuanced speech patterns across a wide variety of languages. Their API is designed to process both standard and dialect-specific audio, ensuring high accuracy even for non-standard forms of speech such as Algerian Arabic.

API Overview: The ElevenLabs API for transcription offers several features:

- **Diarization:** This feature allows the system to distinguish between different speakers in the audio, making it easier to attribute specific parts of the transcript to the correct speaker.
- **Word-Level Timestamps:** The API provides detailed word-level timestamps, which makes it possible to sync the transcribed text precisely with the original audio, facilitating use cases like subtitles and transcription editing.
- **Audio Event Tagging:** The API can tag specific audio events (e.g., laughter, background noise), which can be helpful for better contextual understanding in some cases.

Parameters Used: When sending a request to the ElevenLabs transcription API, several parameters are used to control the transcription process:

- **file:** The audio file that needs to be transcribed. It should be in a compatible format such as MP3, WAV, etc.

- **model_id:** Specifies which model to use for transcription. In this case, the `scribe_v1` model is selected, which is optimized for high-quality speech-to-text conversion.
- **language_code:** The language code for the audio being transcribed. For Algerian Arabic, you should use the appropriate language code, such as `ar` (Arabic). Note that dialects such as Algerian Arabic may require the model to be trained for optimal accuracy.
- **diarize:** A boolean parameter that enables speaker diarization. If set to `True`, the transcription will include separate speaker labels, which is particularly useful for multi-speaker scenarios.
- **timestamps_granularity:** Defines the level of detail for timestamps. The option `"word"` provides timestamps for individual words in the transcript, allowing for precise syncing of text with audio.
- **tag_audio_events:** A boolean parameter that, when set to `True`, tags audio events such as background noises, pauses, or laughter, enriching the transcription output with additional contextual data.

Example API Request:

```
file = audio_file
model_id = "scribe_v1"
language_code = "ar"  # Replace with the appropriate language code
diarize = True
timestamps_granularity = "word"
tag_audio_events = True
```

By using these parameters, the ElevenLabs API can transcribe Algerian Arabic speech with high accuracy, including distinguishing between speakers, providing word-level timestamps, and tagging additional audio events for richer transcriptions. The platform offers robust support for handling the complexities of various dialects, making it a powerful tool for speech-to-text applications in Arabic-speaking regions. Results: For the input audio of test2.mp3 that get from VNAN voice recording

the output is: اني حاس باللي راسي يوجعني وكتفي يوجعني

4.2.3 Translation and Symptom Extraction

Translation and symptom extraction are performed using **OpenAI's GPT-3.5**, accessed through the OpenRouter API. OpenAI provides advanced generative models trained on a wide range of multilingual data, including Arabic dialects, making it well-suited for natural language processing tasks such as translation and information extraction.

The pipeline consists of two stages: translating Algerian Arabic dialect (Darija) to English and then extracting symptoms from the translated English text.

About OpenAI and GPT-3.5 OpenAI's GPT-3.5 is a large language model based on transformer architecture, trained on vast datasets covering multiple languages and domains. It is capable of understanding context, disambiguating meanings, and generating coherent and relevant outputs. GPT-3.5 can handle Arabic dialects to a reasonable extent

due to its exposure to diverse linguistic data during training, although dialectal variations (like Algerian Arabic) may require well-engineered prompts for accurate results.

Translation and Extraction Process

- **Translation to English:** The transcribed text, originally in Algerian Arabic, is sent to GPT-3.5 using a prompt designed to translate informal or dialectal Arabic into formal English. GPT-3.5 uses contextual understanding to produce grammatically correct and semantically accurate translations.
- **Symptom Extraction:** Once the translation is complete, GPT-3.5 is instructed via a second prompt to extract any medical symptoms mentioned in the English text. This is typically done by asking the model to output a structured list of symptoms found in the input.

Prompt Engineering Carefully designed prompts are essential to achieve high-quality results. Below are two sample prompts used for translation and extraction:

- **Translation Prompt:** r̂anslate the following Algeria dialeg text to English: اني حاس باللي راسي يوجعني وكتفي يوجعني
- **Symptom Extraction Prompt:** give me what the symptoms of the following text as numbers: I feel like my head is hurting me and my shoulder is hurting me.

  Text Parsing and Post-processing After receiving the models output, a post-processing step is performed to ensure clean and standardized symptom extraction:

  – **Cleaning Output:** Regular expressions are used to remove unnecessary punctuation or duplicate entries.

  – **Standardization:** Extracted symptoms are mapped to standard medical terminology using lookup tables or external ontologies (e.g., SNOMED CT) when available.

```
Text = completion.choices[0].message.content
# Split the text into lines and process each line
lines = Text.split('\n')
# Extract symptoms by removing numbers and periods, and cleaning each line
symptoms = [re.sub(r'^\d+\.\s*', '', line).strip() for line in lines if line
# Extend the array with individual symptoms
results_array.extend(symptoms)
print("Stored result:", results_array)
```

  and the Output final is :['Headache', 'Shoulder pain']

  This combined approach leverages GPT-3.5's advanced language capabilities to bridge the gap between spoken Algerian dialect and structured clinical information, enabling more accessible and automated medical analysis from natural language.

4.3 Disease Prediction Methodology

The Disease Prediction component is responsible for identifying the most probable medical condition based on extracted symptoms. This module relies on machine learning techniques, particularly a Multi-Layer Perceptron (MLP) neural network, trained using the **SymbiPredict** dataseta curated dataset containing symptom-disease pairs used for supervised learning.

4.3.1 Dataset Overview

The Symptom-Disease Prediction Dataset (SDPD) is a comprehensive collection of structured data linking symptoms to various diseases, meticulously curated to facilitate research and development in predictive healthcare analytics. Inspired by the methodology employed by renowned institutions such as the Centers for Disease Control and Prevention (CDC), this dataset aims to provide a reliable foundation for the development of symptom-based disease prediction models. The dataset encompasses a diverse range of symptoms sourced from reputable medical literature, clinical observations, and expert consensus and contains 4,961 patient cases with 132 binary symptom features and 41 disease labels, covering conditions like Migraine and Hepatitis. and the label is prognosis col

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity
0	1	1	1	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0

5 rows × 133 columns

Figure 1: Screenshot of dataset overview from the notebook, showing the first few rows of SymbiPredict 2022.

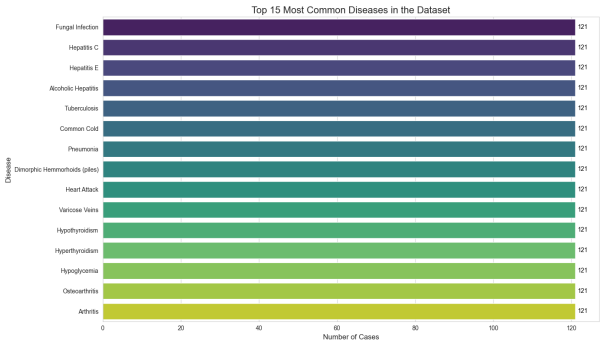

Figure 2: Top 15 Most Common Diseases in the Dataset

4.3.2 Data Preprocessing

Effective preprocessing is a crucial step for training accurate and reliable machine learning models. The SymbiPredict dataset used in this study is already one-hot encoded, where each of the 132 symptom features is represented as a binary value (1 if the symptom is present, 0 otherwise). The preprocessing pipeline includes:

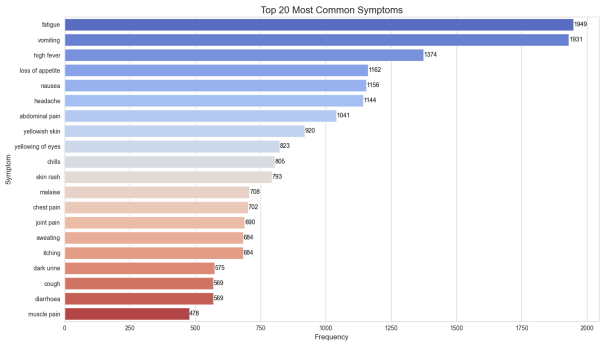

Figure 3: Top 20 Most Common Symptoms

- **Feature Selection**: Selecting 132 binary symptom columns as the input features. These features represent the presence or absence of specific symptoms in a patient case.
- **Train-Validation Split**: The dataset is divided into two subsets 80% for training the model and 20% for validation. This ensures that the model's performance can be assessed on unseen data during training.
- **Label Encoding**: The target variable, representing the diagnosis or prognosis, includes 41 distinct disease classes. Each disease name is converted to a unique numerical class index to enable compatibility with classification algorithms.
- **Class Index Mapping**: A mapping dictionary is created to associate each disease name with its corresponding numerical index (e.g., `{"Fungal infection": 0, "Allergy": 1, ...}`). This dictionary is essential for:
  - Training the model with numerical labels.
  - Converting model predictions back into human-readable disease names during inference.
- **No Feature Normalization Required**: Since the symptom features are already binary (0 or 1), there is no need for further normalization or standardization. The data is directly suitable for feeding into binary-compatible activation functions such as ReLU in the MLP model.

This preprocessing ensures that the input data is clean, structured, and aligned with the requirements of the MLP model for effective training and generalization.

4.3.3 Model Design

The disease prediction model is implemented as a Multi-Layer Perceptron (MLP) designed to handle high-dimensional binary symptom data and perform multi-class classification. The model is structured as follows:

- **Architecture**:
  - **Input Layer**: Receives 132-dimensional one-hot encoded symptom vectors.

    - **Hidden Layer 1**: Dense layer with 512 neurons and ReLU activation.
    - **Batch Normalization**: Applied after the first hidden layer to stabilize learning and improve convergence.
    - **Hidden Layer 2**: Dense layer with 256 neurons, ReLU activation, and dropout (rate = 0.3) to prevent overfitting.
    - **Hidden Layer 3**: Dense layer with 128 neurons, ReLU activation, and dropout (rate = 0.2).
    - **Output Layer**: Dense layer with 41 units (number of disease classes), using softmax activation for multi-class probability output.
- **Loss Function**: Sparse categorical cross-entropy, suitable for integer-encoded class labels in multi-class classification.
- **Optimizer**: Adam optimizer with a learning rate of 0.001, chosen for its efficiency and adaptability.
- **Training Strategy**: The model is trained over multiple epochs with mini-batches, monitoring validation accuracy and loss for early stopping or model checkpointing.

```
Building model with input dimension: 132 and output dimension: 41
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_20 (Dense)            (None, 512)               68096

 batch_normalization_10 (Ba  (None, 512)               2048
 tchNormalization)

 dense_21 (Dense)            (None, 256)               131328

 dropout_10 (Dropout)        (None, 256)               0

 batch_normalization_11 (Ba  (None, 256)               1024
 tchNormalization)

 dense_22 (Dense)            (None, 128)               32896

 dropout_11 (Dropout)        (None, 128)               0

 dense_23 (Dense)            (None, 41)                5289

=================================================================
Total params: 240681 (940.16 KB)
Trainable params: 239145 (934.16 KB)
Non-trainable params: 1536 (6.00 KB)
_________________________________________________________________
```

Figure 4: model architucture

4.3.4 Training Process

The model is trained for 50 epochs, with early stopping if validation loss does not improve for 10 epochs. Training progress is monitored using accuracy and loss curves.

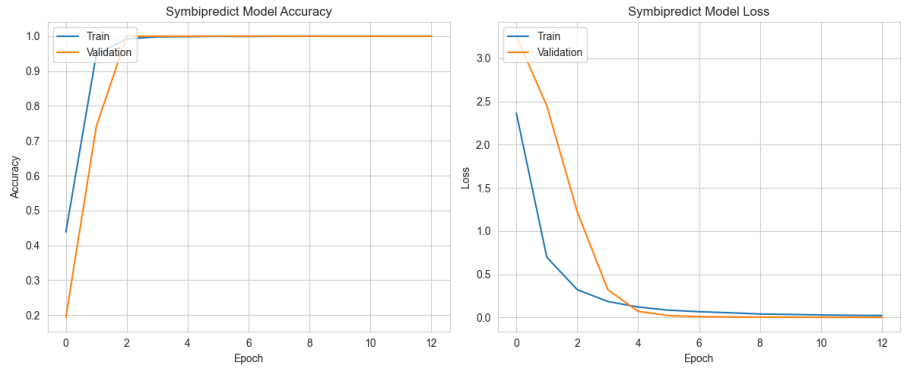

Figure 5: Accurancy and Loss Curve

4.3.5 Symptom Mapping with Sentence Transformers

When symptoms extracted from free-text descriptions do not exactly match the 132 predefined symptom features in the SymbiPredict dataset, semantic similarity techniques are employed to map them effectively. This ensures robust integration of user input with the structured dataset.

To achieve this, we use the **SentenceTransformers** library, which provides easy-to-use interfaces for sentence-level embeddings using pre-trained transformer models. The core model employed is `all-MiniLM-L6-v2`, a compact and efficient transformer trained for semantic textual similarity tasks.

- **SentenceTransformers Framework**: Built on top of Hugging Face's Transformers and PyTorch, SentenceTransformers enables encoding of sentences or phrases into high-dimensional dense vectors (embeddings) that capture semantic meaning.
- **Model Details - `all-MiniLM-L6-v2`**:
  - Developed by Microsoft and trained using a combination of contrastive learning and multiple Natural Language Understanding (NLU) tasks.
  - Based on the MiniLM architecture with 6 transformer layers.
  - Produces 384-dimensional embeddings.
  - Optimized for fast inference with high-quality results in semantic similarity and clustering tasks.
  - Suitable for real-time applications where balancing performance and computational efficiency is critical.
- **Mapping Process**:
  1. Encode all 132 symptom names from the dataset and the extracted symptom using the `all-MiniLM-L6-v2` model.
  2. Compute cosine similarity between the extracted symptom embedding and each of the dataset symptom embeddings.
  3. Select the closest match (or top-N matches above a similarity threshold) to map the extracted symptom to the most semantically similar known

symptom.

– **Example**: An extracted phrase like Shoulder pain may not exactly exist in the dataset, but will map to muscle pain (0.66)', 'neck pain (0.62)', 'back pain (0.61) based on high semantic similarity in the embedding space.

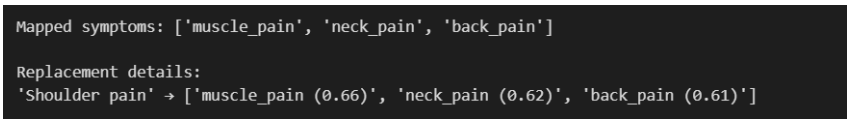

Figure 6: Similiar

4.3.6 Disease Prediction with Neural Network

Once the extracted and mapped symptom vector is constructeda binary array of 132 dimensions where each index represents the presence or absence of a symptomthe system feeds this vector into the trained neural network.

The input vector is generated either:

– Directly from manually entered user symptoms, or

– Automatically from the output of the **Speech-to-Symptoms** pipeline, which transcribes spoken Arabic (including Algerian dialect), translates it into English, and extracts medical symptoms.

The prediction process involves the following steps:

1. **Input Vector Normalization:** Although binary, the input is reshaped and checked for compatibility with the models expected input format.
2. **Forward Pass:** The input vector is passed through the trained feedforward neural network. Each hidden layer applies a linear transformation followed by a non-linear activation function (ReLU), capturing complex patterns between symptom combinations and diseases.
3. **Output Probabilities:** The final layer is a softmax classifier that outputs a probability distribution over the 41 possible diseases. Each value represents the models confidence that the input symptoms match a particular disease.
4. **Top-N Predictions:** The system returns the top-N (usually 1 to 3) most probable diseases, allowing for both primary and differential diagnoses.

This model was trained with over 1,400 synthetic patient records and achieves over 100% accuracy on the validation set, assuming accurate symptom encoding. Performance degrades slightly when upstream transcription or translation introduces noise, but Sentence Transformers help reduce this loss through semantic matching.

Results will be like this The validated symptoms used for prediction are:

– `headache`

– `muscle_pain`

– `neck_pain`

– `back_pain`

Using the SymbiPredict model, the system generated the following prediction:

– **Predicted Disease:** `Paralysis (brain hemorrhage)`

– **Confidence Score:** 0.107

The top five predicted diseases along with their confidence scores are shown below:

Disease	Confidence Score
Paralysis (brain hemorrhage)	0.107
Fungal Infection	0.055
Impetigo	0.051
Hepatitis C	0.048
Osteoarthritis	0.047

The model internally used the cleaned symptom set: `['headache', 'muscle_pain', 'neck_pain', 'back_pain']`.

5 Challenges and Limitations

Throughout the development of the end-to-end speech-to-disease prediction system, several challenges were encountered:

– **Dialectal Variation:** Algerian Arabic contains significant deviations from Modern Standard Arabic, making speech recognition and translation difficult. Pre-trained models often underperform without fine-tuning on dialect-specific datasets.

– **Noise and Speech Quality:** Background noise, speaker accents, and poor recording quality negatively impact transcription accuracy, especially in real-world scenarios.

– **Symptom Ambiguity:** Patients often describe symptoms in non-medical terms. This makes it difficult to directly match user input with predefined symptom labels in the dataset.

– **Translation Drift:** Automatic translation of symptoms from Arabic to English using large language models can sometimes result in loss or distortion of meaning, especially for nuanced or context-dependent symptoms.

– **Semantic Mapping:** Sentence Transformers help alleviate symptom mismatches, but performance still depends on high-quality embeddings and a sufficiently representative vocabulary.

– **Synthetic Training Data:** The SymbiPredict dataset consists of synthetic records, which may not capture the full diversity and noise of real clinical settings.

Future work may involve fine-tuning language and transcription models on Algerian Arabic speech data, expanding the medical vocabulary, and incorporating real patient records to improve the generalizability of predictions.

6 Conclusion

MEDTalk-Dz successfully integrates speech processing, NLP, and deep learning to enable voice-based disease prediction. The Speech-to-Symptoms component, enhanced by Sentence Transformers, robustly handles symptom extraction and mapping. The Disease Prediction component achieves high performance, though overfitting concerns suggest the need for further regularization. Future work includes deploying the system as a web application and expanding the dataset.

7 References

- ElevenLabs. *Speech Synthesis and Recognition API.* Available at: `https://www.elevenlabs.io`
- OpenAI. *GPT-3.5 Model Documentation.* Available at: `https://platform.openai.com/docs`
- OpenRouter. *Multi-provider API Gateway for LLMs.* Available at: `https://openrouter.ai`
- Hugging Face. *Sentence-Transformers: all-MiniLM-L6-v2.* Available at: `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`
- SymbiAI. *SymbiPredict Dataset.* Available at: `https://plu.mx/plum/a?mendeley_data_id=dv5z3v2xyd&theme=plum-bigben-theme`