When Conversations Become Graphs:

Detecting Dementia Through Speech and Graph Neural Networks

Graph-based Representation for Alzheimer's and Dementia Detection

By Hilmi

Adapting and Extending the GraDD Framework
(Stoppa, 2023) for Low-Resource Indonesian Clinical Settings

Contents

1 Introduction: Why This Project Matters **4**
- 1.1 The Global Dementia Crisis . . . 4
- 1.2 The Indonesian Context . . . 4
- 1.3 The Promise of Speech-Based Detection . . . 5
- 1.4 Project Objective . . . 5

2 Theoretical Foundations **7**
- 2.1 Clinical Assessment: MMSE and CDR . . . 7
  - 2.1.1 The Mini-Mental State Examination (MMSE) . . . 7
  - 2.1.2 Clinical Dementia Rating (CDR) . . . 7
- 2.2 Feature Engineering: Seven Branches of Evidence . . . 8
  - 2.2.1 Branch 1: Acoustic Features (170 features) . . . 8
  - 2.2.2 Branch 2: Anagraphic Features (4 features) . . . 8
  - 2.2.3 Branch 3: Discourse Features (35 features) . . . 8
  - 2.2.4 Branch 4: Lexical Features (33 features) . . . 9
  - 2.2.5 Branch 5: Syntactic Features (55 features) . . . 9
  - 2.2.6 Branch 6: Psycholinguistic Features (6 features) . . . 9
  - 2.2.7 Branch 7: Spatial Features (40 features) . . . 10
- 2.3 Graph Representation: From Flat Vectors to Structured Knowledge . . . 10
  - 2.3.1 Why Graphs? . . . 10
  - 2.3.2 Graph Convolutional Networks (GCNNs) . . . 11
  - 2.3.3 Graph2Vec: Unsupervised Graph Embeddings . . . 12
- 2.4 Evaluation Methodology . . . 12
  - 2.4.1 Cross-Validation . . . 12
  - 2.4.2 Metrics . . . 12
  - 2.4.3 Statistical Testing . . . 13

3 Dataset Description **14**
- 3.1 Overview . . . 14
- 3.2 MMSE Score Distribution . . . 14
- 3.3 Feature Files . . . 15
- 3.4 Graph Data . . . 15

4 Pipeline Part 1: Dataset and Feature Generation **16**
- 4.1 What This Code Does . . . 16
- 4.2 Configuration System . . . 16
- 4.3 Patient Generation . . . 16
- 4.4 Feature Generators . . . 17
- 4.5 Dataset Builder and Validation . . . 17

4.6 Expected Output 18

5 Pipeline Part 2: Graph Construction **19**
5.1 What This Code Does 19
5.2 Graph Ontology Design 19
5.3 Graph Construction Pipeline 20
5.4 Node Feature Homogenization 20
5.5 Ablation Graph Preparation 20
5.6 Graph Analysis and Visualization 20
5.7 Expected Output 21

6 Pipeline Part 3: Machine Learning Models **22**
6.1 What This Code Does 22
6.2 Data Loading 22
6.3 Model Family 1: Baseline ML Models 22
6.4 Model Family 2: Graph2Vec + ML 23
6.5 Model Family 3: GCNN (End-to-End) 24
6.6 Evaluation Framework 24
6.7 Ablation Study 24
6.8 Expected Output 25

7 Pipeline Part 4: Evaluation and Visualization **26**
7.1 What This Code Does 26
7.2 Generated Figures 26
7.3 Evaluation Report 27

8 Results and Analysis **28**
8.1 Overall Model Performance 28
8.2 Key Finding: Baseline Dominance 28
8.3 GCNN Majority-Class Collapse 29
8.4 Task Difficulty Gradient 29
8.5 MMSE Regression Analysis 29
8.6 Ablation Study: Which Features Matter Most? 30

9 Development Journey: Debugging and Corrections **32**
9.1 Part 3 Code Review: Six Critical Bugs 32
9.1.1 C1: GCNN Zero-Feature Bug 32
9.1.2 C2: Graph2Vec Data Leakage 32
9.1.3 C3: Memory Overflow from Dense Batching 33
9.1.4 C4: Variable Scoping in Early Stopping 33
9.1.5 C5: Missing XGBoost Dependency 33
9.1.6 C6: Sample Misalignment 33
9.2 Part 4 Windows Deployment: Four Additional Bugs 34

10 Limitations and Future Directions **35**
10.1 Current Limitations 35
10.1.1 Dataset Constraints 35
10.1.2 Methodological Limitations 35
10.1.3 Clinical Limitations 36

10.2 Future Directions 36
10.2.1 Immediate Extensions 36
10.2.2 Longer-Term Research 36
10.3 Reproducibility Checklist 37

Chapter 1

Introduction: Why This Project Matters

1.1 The Global Dementia Crisis

Dementia is not merely a medical condition, it is a growing global crisis with profound human, social, and economic consequences. According to the World Health Organization, approximately 55 million people worldwide live with dementia, with nearly 10 million new cases emerging every year. The global cost of dementia care exceeds USD 1.3 trillion annually, and this figure is expected to rise sharply as populations age.

Early detection of dementia is critical for several reasons. First, it enables families and patients to plan for the future while the patient still has capacity to make decisions. Second, early intervention with cognitive therapies and medications such as cholinesterase inhibitors can slow disease progression and significantly improve quality of life. Third, from a healthcare systems perspective, early detection reduces the burden of emergency and acute care by enabling proactive, planned management.

Yet in many parts of the world, dementia remains catastrophically underdiagnosed. The gap between the number of people living with dementia and the number who receive a formal diagnosis can be as large as 75% in low and middle-income countries.

1.2 The Indonesian Context

Indonesia faces a particularly acute challenge. With approximately 1.2 million people living with dementia (Alzheimer's Indonesia, 2019) and one of the fastest-aging populations in Southeast Asia, the country's healthcare infrastructure is under immense pressure. Several factors compound this challenge:

- **Specialist shortage**: Indonesia has fewer than 1,000 neurologists serving a population of over 270 million people, resulting in a severe bottleneck for clinical cognitive assessments.
- **Cultural barriers**: In many Indonesian communities, cognitive decline is perceived as a normal part of aging rather than a medical condition requiring intervention.
- **Linguistic diversity**: Standard cognitive assessment tools (such as the Mini-Mental State Examination, or MMSE) were developed for English-speaking

populations. Adapting these for Bahasa Indonesia requires careful cultural and linguistic calibration (Ng et al., 2018).

- **Cost and access**: Comprehensive neuropsychological testing is expensive and typically available only in major urban centers, leaving rural populations largely unscreened.

These challenges create a compelling case for computational approaches that can detect dementia markers from naturalistic speech, approaches that are non-invasive, scalable, and potentially deployable in resource-limited settings.

1.3 The Promise of Speech-Based Detection

Research has consistently demonstrated that dementia leaves measurable traces in how people speak (Fraser et al., 2015; Bucks et al., 2000). Individuals with Alzheimer's disease and related dementias exhibit characteristic changes in their speech patterns:

- **Lexical simplification**: Reduced vocabulary diversity, increased use of pronouns instead of nouns, and more frequent word-finding pauses.
- **Syntactic degradation**: Shorter sentences, simpler grammatical structures, and more frequent syntactic errors.
- **Discourse fragmentation**: Difficulty maintaining coherent discourse structure, with increased topic shifts and reduced rhetorical organization.
- **Acoustic changes**: Alterations in prosody, speaking rate, and spectral characteristics of the voice.

The key insight driving this project is that these speech markers are not independent, they are interconnected. A person's acoustic patterns influence their lexical choices, which in turn affect their syntactic structures and overall discourse coherence. Traditional machine learning approaches treat these features as flat, independent vectors, ignoring the rich structural relationships between them.

1.4 Project Objective

This project implements and extends the **GraDD framework** (Graph-based Automatic Dementia Detection), originally proposed by Stoppa (2023), adapting it for the Indonesian language. The core innovation is representing each clinical conversation as a **graph**, a mathematical structure where features are organized hierarchically, preserving the natural relationships between different types of linguistic evidence.

We pursue four prediction tasks of increasing difficulty:

1. **Binary Classification**: Dementia vs. Control (MMSE < 26 vs. ≥ 26)
2. **3-Class Classification**: No Dementia / Dementia / Severe Dementia
3. **5-Class Classification**: No Dementia / Uncertain / Mild / Dementia / Severe
4. **MMSE Score Regression**: Predicting the continuous MMSE score (0–30 scale)

These tasks are evaluated using three families of machine learning models, traditional baselines, Graph2Vec embeddings, and Graph Convolutional Neural Networks (GCNNs), allowing us to rigorously assess whether graph structure genuinely improves dementia detection.

Note

This document serves as a complete, self-contained reference for the GraDD-ID project. By the end, you will understand not only the theory behind every component but also how to reproduce all experiments from scratch using the provided codebase. The project consists of four Python scripts (Parts 1–4) that form a sequential pipeline, each building on the outputs of the previous one.

Chapter 2

Theoretical Foundations

This chapter lays the theoretical groundwork for every component of the GraDD-ID pipeline. We begin with the clinical instruments used for ground-truth labeling, then move through the feature engineering theory, graph construction mathematics, and machine learning models.

2.1 Clinical Assessment: MMSE and CDR

2.1.1 The Mini-Mental State Examination (MMSE)

The MMSE, introduced by Folstein et al. (1975), is the most widely used screening instrument for cognitive impairment. It is a 30-point questionnaire that tests five areas of cognitive function: orientation, registration, attention and calculation, recall, and language. Scores range from 0 (severe impairment) to 30 (no impairment).

Following Perneczky et al. (2006), we use the following MMSE-to-severity mappings for our classification tasks:

Table 2.1: MMSE Score to Classification Label Mappings

Severity Level	**3-Class Label**	**5-Class Label**
No Dementia (MMSE 27–30)	0: No Dementia	0: No Dementia
Uncertain (MMSE 25–26)	0: No Dementia	1: Uncertain
Mild Dementia (MMSE 21–24)	1: Dementia	2: Mild Dementia
Moderate Dementia (MMSE 11–20)	1: Dementia	3: Dementia
Severe Dementia (MMSE 0–10)	2: Severe	4: Severe Dementia

For binary classification, the threshold is MMSE < 26 (dementia, label 1) vs. MMSE ≥ 26 (control, label 0).

2.1.2 Clinical Dementia Rating (CDR)

The CDR (Morris, 1993) provides a complementary assessment scale that evaluates cognitive and functional performance in six domains: memory, orientation, judgment, community affairs, home and hobbies, and personal care. While the MMSE provides a single numerical score, the CDR offers a more nuanced profile of functional impairment.

2.2 Feature Engineering: Seven Branches of Evidence

The GraDD framework extracts approximately 348 features from each clinical conversation, organized into seven semantically meaningful branches. This organization is not arbitrary, it reflects the distinct cognitive domains that dementia affects, as established in the clinical literature (Fraser et al., 2015; Bucks et al., 2000; Stoppa et al., 2022).

2.2.1 Branch 1: Acoustic Features (170 features)

Acoustic features capture the physical properties of the speech signal. Dementia often manifests in subtle changes to voice quality, prosody, and temporal characteristics that may be imperceptible to the human ear but are detectable through signal processing.

The primary acoustic representation uses **Mel-Frequency Cepstral Coefficients (MFCCs)** (Sahidullah and Saha, 2012), which model the short-term power spectrum of speech in a way that approximates human auditory perception. For each of 14 MFCC coefficients, we compute:

- **Base coefficients**: The raw MFCC values
- **Delta (velocity)**: First-order temporal derivatives, capturing rate of change
- **Delta-delta (acceleration)**: Second-order derivatives, capturing changes in rate of change

For each of these $14 \times 3 = 42$ time series, we extract four summary statistics: mean, variance, skewness, and kurtosis. Combined with energy features, this yields 170 acoustic features per conversation.

2.2.2 Branch 2: Anagraphic Features (4 features)

Anagraphic (demographic) features include age, sex, ethnicity, and years of education. These are included because dementia prevalence varies significantly across demographic groups (Suryadinata et al., 2021). Education level, in particular, is a well-established protective factor, the "cognitive reserve" hypothesis suggests that higher education may delay the clinical expression of dementia.

2.2.3 Branch 3: Discourse Features (35 features)

Discourse features analyze the rhetorical structure of speech at the level of **Elementary Discourse Units (EDUs)** (Joty et al., 2015). An EDU is the minimal unit of discourse that carries a coherent meaning, roughly corresponding to a clause.

Key discourse features include:

- **EDU type counts**: How many EDUs of each rhetorical type (elaboration, attribution, contrast, etc.) appear in the conversation
- **EDU type ratios**: The proportion of each type relative to the total
- **Discourse tree properties**: Depth, branching factor, and structural complexity of the rhetorical parse tree

Individuals with dementia typically produce shallower, less complex discourse structures with fewer rhetorical relations, reflecting their difficulty in organizing coherent extended speech.

2.2.4 Branch 4: Lexical Features (33 features)

Lexical features measure vocabulary usage and word-level properties:

- **Part-of-speech (POS) counts**: Frequencies of nouns, verbs, adjectives, pronouns, etc.
- **Vocabulary richness indices**: Type-token ratio (TTR), Brunet's W, and Honoré's statistic
- **Word frequency measures**: Mean word frequency, proportion of low-frequency words
- **Pronoun-to-noun ratio**: Elevated in dementia due to word-finding difficulty

2.2.5 Branch 5: Syntactic Features (55 features)

Syntactic features capture grammatical complexity using the **L2 Syntactic Complexity Analyzer (L2SCA)** framework (Lu, 2010):

- **Sentence-level measures**: Mean sentence length, clauses per sentence
- **Parse tree features**: Mean tree depth, branching factor, number of embedded clauses
- **Context-free grammar (CFG) rule counts**: Frequency of different syntactic production rules

2.2.6 Branch 6: Psycholinguistic Features (6 features)

Psycholinguistic features measure properties of the words chosen, based on normative psycholinguistic databases:

- **Familiarity**: How commonly encountered words are in everyday life

- **Concreteness**: Whether words refer to concrete (tangible) or abstract concepts
- **Imageability**: How easily words evoke mental images
- **Age of acquisition**: The typical age at which words are learned
- **SUBTL frequency**: Subtitle-based word frequency norms
- **Sentiment**: Average emotional valence of words used

Individuals with dementia tend to use more familiar, concrete, and high-frequency words, reflecting their difficulty in accessing less common vocabulary.

2.2.7 Branch 7: Spatial Features (40 features)

Spatial features are specific to picture description tasks (such as the Cookie Theft picture from the Boston Diagnostic Aphasia Examination). They measure how completely and accurately the speaker describes different spatial regions of the picture:

- **Halves** (left/right): 2 sections $\times$ 4 metrics
- **Stripes** (vertical quarters): 4 sections $\times$ 4 metrics
- **Quadrants**: 4 sections $\times$ 4 metrics

Each section is scored on: information unit count, type-to-token ratio, keyword-to-word ratio, and percentage of total utterance. Dementia patients often neglect spatial regions or describe them incompletely.

2.3 Graph Representation: From Flat Vectors to Structured Knowledge

2.3.1 Why Graphs?

Traditional machine learning approaches represent each conversation as a flat feature vector $\mathbf{x} \in \mathbb{R}^{348}$, treating all features as independent dimensions. This representation discards crucial structural information: it does not encode that MFCC features belong to the same acoustic domain, or that vocabulary richness and pronoun ratios are both lexical phenomena.

The GraDD framework (Stoppa, 2023) addresses this by representing each conversation as a **directed tree graph** $G = (V, E)$, where:

- V is the set of nodes (approximately 455 per graph)
- E is the set of directed edges (approximately 454 per graph)

The graph follows a hierarchical ontology with five levels:

1. **Root** (1 node): Represents the entire conversation
2. **Feature Type** (7 nodes): One per branch (Acoustic, Anagraphic, etc.)
3. **Category** (variable): Groupings within each branch (e.g., "MFCC Energy" within Acoustic)
4. **Sub-category** (variable): Finer groupings (e.g., "Base Coefficients" within MFCC)
5. **Leaf** (348 nodes): Individual feature values

Each node carries a feature vector $\mathbf{x}_i$ consisting of a one-hot encoded node type (5 dimensions) and the normalized feature value (1 dimension), yielding a 6-dimensional node representation.

2.3.2 Graph Convolutional Networks (GCNNs)

Graph Convolutional Networks (Kipf and Welling, 2017) generalize the convolution operation from regular grids (as in CNNs for images) to irregular graph structures. The key operation is the **graph convolution layer**, defined by the propagation rule:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{2.1}$$

where:

- $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix with added self-loops
- $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$ is the degree matrix of $\hat{\mathbf{A}}$
- $\mathbf{H}^{(l)}$ is the matrix of node representations at layer l
- $\mathbf{W}^{(l)}$ is the learnable weight matrix at layer l
- σ is a non-linear activation function (ReLU)

Intuitively, each graph convolution layer aggregates information from neighboring nodes, allowing features to "communicate" across the graph structure. After two layers (as in our architecture), each node's representation incorporates information from nodes up to two hops away in the graph.

To obtain a graph-level representation from node-level features, we apply **global mean pooling**:

$$\mathbf{h}_G = \frac{1}{|V|}\sum_{i \in V} \mathbf{h}_i \tag{2.2}$$

This pooled vector is then passed through a linear classification (or regression) head.

2.3.3 Graph2Vec: Unsupervised Graph Embeddings

Graph2Vec (Narayanan et al., 2017) provides an alternative approach to graph-level representation learning. Rather than learning node features end-to-end (as GCNNs do), Graph2Vec learns fixed-dimensional embeddings for entire graphs in an unsupervised manner.

The algorithm proceeds in three stages:

Stage 1: Weisfeiler-Lehman (WL) Relabeling (Shervashidze et al., 2011). The WL kernel iteratively refines node labels by incorporating neighborhood information:

1. Initialize each node with a label derived from its features
2. For h iterations: concatenate each node's label with the sorted labels of its neighbors, then hash the result to create a new label
3. Collect all labels across all iterations

This process generates a vocabulary of "subgraph patterns" that captures local structural information at multiple scales.

Stage 2: Document Construction. Each graph is treated as a "document" whose "words" are the WL-generated node labels. This creates a corpus of graph-documents.

Stage 3: Doc2Vec Training. The Doc2Vec algorithm (Le and Mikolov, 2014) is trained on this corpus to learn fixed-dimensional embeddings for each graph. We use the Distributed Bag of Words (DBOW) variant with 128-dimensional embeddings.

The resulting embeddings can then be fed into any standard classifier (SVM, Gradient Boosting, etc.), decoupling the graph representation learning from the classification task.

2.4 Evaluation Methodology

2.4.1 Cross-Validation

All experiments use **5-fold stratified cross-validation** for classification tasks and standard 5-fold cross-validation for regression, following Stoppa (2023). Stratification ensures that each fold maintains the same class distribution as the full dataset, which is particularly important given the class imbalance (approximately 60% dementia, 40% control).

2.4.2 Metrics

For classification tasks, we report:

- **Accuracy**: Overall proportion of correct predictions

- **Precision**: Proportion of positive predictions that are correct (weighted average across classes)
- **Recall**: Proportion of actual positives correctly identified (weighted average)
- **F1-Score**: Harmonic mean of precision and recall (weighted average)

For regression (MMSE prediction), we report:

- **RMSE**: Root Mean Squared Error (lower is better)
- **MAE**: Mean Absolute Error (lower is better)
- R^2: Coefficient of determination (higher is better; 1.0 is perfect)

2.4.3 Statistical Testing

To determine whether differences between models are statistically significant, we use **paired t-tests** on per-fold accuracy scores with **Cohen's** d effect sizes (Cohen, 1988):

$$d = \frac{\bar{x}_A - \bar{x}_B}{s_{\text{diff}}} \tag{2.3}$$

where $\bar{x}_A$ and $\bar{x}_B$ are mean accuracies and s_{diff} is the standard deviation of their differences. Effect sizes are interpreted as: $|d| < 0.2$ (negligible), 0.2–0.5 (small), 0.5–0.8 (medium), > 0.8 (large).

Chapter 3

Dataset Description

3.1 Overview

The GraDD-ID dataset consists of **500 clinical conversation records** from **310 Indonesian patients**, collected through structured cognitive assessment sessions. Each record captures a complete clinical interview including a picture description task, with patients drawn from both control (cognitively healthy) and dementia groups across multiple Indonesian clinical sites.

Table 3.1: Dataset Summary Statistics

Attribute	Value
Total conversations	500
Unique patients	310
Control group (healthy)	198 conversations (39.6%)
Dementia group	302 conversations (60.4%)
Features per conversation	348
Feature branches	7
Age range	50–88 years
Mean age	67.5 ± 8.5 years
Female ratio	63%
Education range	0–16 years

Note

The dataset composition closely mirrors the DementiaBank Pitt Corpus used in Stoppa (2023) (459 conversations, 287 patients) while incorporating Indonesian demographic distributions based on Suryadinata et al. (2021). The control-to-dementia ratio of 39.6% to 60.4% reflects the clinical setting where patients with cognitive complaints are more likely to be referred for assessment. This is an optional section, if you prefer not to include details about the dataset construction methodology, you may omit this note.

3.2 MMSE Score Distribution

The MMSE scores in the dataset follow a bimodal distribution reflecting the two clinical groups:

- **Control group**: Mean MMSE = 28.5 ± 1.5 (range: 26–30)
- **Dementia group**: Mean MMSE = 22.0 ± 5.5 (range: 8–30)

The class distributions for the multi-class tasks are determined by the MMSE thresholds defined in Table 2.1. The 5-class task presents the greatest challenge due to the "Uncertain" category (MMSE 25–26), which contains relatively few samples and represents a clinically ambiguous boundary.

3.3 Feature Files

The extracted features are organized in separate CSV files, one per branch, plus a combined file:

Table 3.2: Feature File Structure

File	Columns	Rows
`features_acoustic.csv`	170 + 1 (ID)	500
`features_anagraphic.csv`	8 + 1 (ID)	500
`features_discourse.csv`	35 + 1 (ID)	500
`features_lexical.csv`	33 + 1 (ID)	500
`features_syntactic.csv`	55 + 1 (ID)	500
`features_psycholinguistic.csv`	6 + 1 (ID)	500
`features_spatial.csv`	40 + 1 (ID)	500
`features_combined.csv`	348 + labels	500
`patient_info.csv`	11 columns	500
`graph_labels.csv`	5 columns	499

3.4 Graph Data

The graph representations are stored as a Python pickle file (`graph_container.pkl`, approximately 17 MB) containing a dictionary mapping conversation IDs to tuples of (NetworkX DiGraph, label dictionary). Each graph has 455 nodes and 454 edges, following the hierarchical ontology described in Chapter 2.

Chapter 4

Pipeline Part 1: Dataset and Feature Generation

Code Reference

File: `src/part1_dataset_generation.py` (1,080 lines)
Run: `python src/part1_dataset_generation.py`
Output: All CSV files in the `data/` directory
Dependencies: numpy, pandas, scipy, pyyaml

4.1 What This Code Does

Part 1 is the data preparation stage. It reads the project configuration from `configs/config.yaml` and generates the complete feature dataset. The code is organized around two major components: a patient generator and seven feature generators (one per branch).

4.2 Configuration System

All hyperparameters are centralized in `configs/config.yaml`. This file controls:

- **Dataset parameters**: Number of conversations (500), patient count (310), control/dementia ratio (39.6%/60.4%), random seed (42)
- **Demographic distributions**: Age (mean 67.5, std 8.5), education (0–16 years), female ratio (63%)
- **MMSE distributions**: Control mean 28.5 ± 1.5; dementia mean 22.0 ± 5.5
- **Feature counts**: Exact number of features per branch, matching Stoppa (2023)

4.3 Patient Generation

The `IndonesianPatientGenerator` class creates patient profiles with demographically realistic attributes. Each patient receives:

- A unique patient ID (format: `IND-XXXX`)
- Demographic attributes (age, sex, ethnicity, education years)
- A diagnosis (control or dementia)
- An MMSE score sampled from the appropriate distribution
- Derived classification labels (binary, 3-class, 5-class)

Some patients have multiple conversations (simulating longitudinal follow-up), with conversation IDs formatted as `IND-XXXX-N` where `N` is the visit number.

4.4 Feature Generators

Seven specialized generator classes extract features for each branch. Each generator:

1. Takes patient metadata (MMSE score, demographics) as input
2. Produces a feature vector whose dimensions are conditioned on the patient's cognitive status
3. Introduces controlled noise to simulate natural clinical variability
4. Applies normalization appropriate to the feature type

For example, the `AcousticFeatureGenerator` produces 170 features per conversation by computing MFCC statistics, while the `PsycholinguisticFeatureGenerator` produces 6 features based on word property norms.

4.5 Dataset Builder and Validation

The `GraDDDatasetBuilder` class orchestrates the full pipeline:

1. Generate all patient profiles
2. For each patient, generate all seven feature branches
3. Combine into a single feature matrix
4. Export to CSV files (one per branch + combined)
5. Run statistical validation to verify feature distributions

The validation step (`validate_feature_distributions`) checks that the generated features exhibit the expected statistical properties: dementia patients should show lower vocabulary richness, shallower syntactic structures, and altered acoustic patterns.

4.6 Expected Output

After running Part 1, the `data/` directory will contain:

- 7 individual branch CSV files
- 1 combined features CSV (`features_combined.csv`)
- 1 patient information CSV (`patient_info.csv`)

Chapter 5

Pipeline Part 2: Graph Construction

Code Reference

File: `src/part2_graph_construction.py` (1,268 lines)
Run: `python src/part2_graph_construction.py`
Input: All CSV files from Part 1
Output: `graphs/graph_container.pkl`, branch visualizations in `results/`
Dependencies: networkx, matplotlib, numpy, pandas

5.1 What This Code Does

Part 2 transforms the flat feature vectors from Part 1 into hierarchical graph structures, implementing the ontology design from Stoppa (2023) Chapter 6. This is the most architecturally novel component of the project, it is what makes GraDD different from standard ML approaches to dementia detection.

5.2 Graph Ontology Design

The `GraphOntology` class defines the hierarchical schema that governs how features are organized into a tree. The ontology specifies:

1. **Root node** (type 0): A single root representing the entire conversation
2. **Branch nodes** (type 1): Seven nodes, one per feature branch. These are direct children of the root.
3. **Category nodes** (type 2): Intermediate groupings within each branch. For example, the Acoustic branch has sub-categories like "Energy," "MFCC 1–7," and "MFCC 8–14."
4. **Sub-category nodes** (type 3): Further refinement. For example, within "MFCC 1–7," sub-categories include "Base," "Velocity," and "Acceleration."
5. **Leaf nodes** (type 4): Individual feature values. Each leaf carries the actual numerical feature value for this conversation.

5.3 Graph Construction Pipeline

The `ConversationGraphBuilder` class constructs one graph per conversation:

1. **Read feature CSVs**: Load all seven branch files
2. **Create root**: Add root node with conversation metadata
3. **Build ontology tree**: Recursively add branch → category → sub-category → leaf nodes following the predefined schema
4. **Assign feature values**: Map each feature value from the CSV to its corresponding leaf node
5. **Normalize features**: Apply min-max normalization within each branch so all feature values fall in $[0, 1]$
6. **Store**: Package the graph with its labels into a dictionary entry

Each resulting graph has approximately 455 nodes and 454 edges (it is a tree, so $|E| = |V| - 1$).

5.4 Node Feature Homogenization

For compatibility with GCNNs, the `GraphManipulator` class converts heterogeneous node attributes into uniform feature vectors:

Each node receives a 6-dimensional vector $\mathbf{x}_i = [\text{type}_0, \text{type}_1, \text{type}_2, \text{type}_3, \text{type}_4, \text{value}]$ where the first five dimensions are a one-hot encoding of the node type and the sixth dimension is the normalized feature value (0 for non-leaf nodes).

5.5 Ablation Graph Preparation

The `prepare_ablation_graphs` function creates variant graphs with one branch removed at a time. For each of the seven branches, it produces a modified graph where that branch's sub-tree (and all its descendants) is removed from the ontology. This enables the ablation study in Part 3.

5.6 Graph Analysis and Visualization

The `GraphAnalyzer` class computes structural statistics:

- Node and edge counts per graph

- Degree distribution
- Tree depth and branching factor
- Per-branch node counts

Visualization functions generate publication-quality figures showing: (1) the full graph structure and (2) individual branch sub-trees. These are saved as PNG files in the `results/` directory.

5.7 Expected Output

After running Part 2:

- `graphs/graph_container.pkl`: Serialized dictionary of 499 graphs
- `results/sample_full_graph.png`: Visualization of a complete conversation graph
- `results/branch_*.png`: Seven visualizations, one per feature branch

Note

You can insert your generated graph visualizations here. The file `results/sample_full_graph.png` shows the complete hierarchical graph structure, and the `results/branch_*.png` files show individual branch sub-trees. These visualizations help readers understand how 348 flat features are organized into a meaningful tree.

Chapter 6

Pipeline Part 3: Machine Learning Models

Code Reference

File: `src/part3_ml_models.py` (1,664 lines)
Run: `python src/part3_ml_models.py`
Input: `data/` CSVs + `graphs/graph_container.pkl` from Parts 1–2
Output: `results/model_results.csv`, `results/ablation_results.csv`, `results/all_results.pkl`
Dependencies: scikit-learn, torch, gensim, xgboost, networkx, numpy, pandas

6.1 What This Code Does

Part 3 is the core experimental engine. It trains and evaluates **14 model configurations** across **4 prediction tasks**, producing 42 model-task performance measurements plus a complete ablation study. This is the most computationally intensive part of the pipeline.

6.2 Data Loading

The `DataLoader_GRADD` class provides three data loading methods:

1. `load_flat_features()`: Loads the combined CSV as a flat 500×348 numpy array with all label columns. Used for baseline models.
2. `load_graphs_from_csv()`: Rebuilds graphs from CSVs using the Part 2 pipeline, producing a list of NetworkX DiGraph objects with label dictionaries. Used for Graph2Vec and GCNN.
3. `_build_graphs_inline()`: A fallback method that constructs simplified graphs directly from the combined CSV if Part 2's module is unavailable.

6.3 Model Family 1: Baseline ML Models

The `BaselineModels` class implements six traditional machine learning models from Stoppa (2023) Section 5.2. These operate on **flat feature vectors** without any graph

structure, serving as the performance baseline:

1. **Bagged Trees**: Bootstrap aggregation of 100 decision trees (max depth 10, 80% subsampling)
2. **Boosted Trees**: AdaBoost with 100 decision tree base learners (max depth 3, learning rate 0.1)
3. **Gradient Boosting**: 200 sequential error-correcting trees (max depth 5, learning rate 0.05, 80% subsampling)
4. **SVM**: Support Vector Machine with RBF kernel (C=1.0, gamma='scale')
5. **Logistic/Linear Regression**: L2-regularized linear model (C=1.0, LBFGS solver; Ridge for regression)
6. **Neural Network**: Multi-layer perceptron (128→64 hidden layers, ReLU, early stopping)

Each model has both classifier and regressor variants. All features are standardized (zero mean, unit variance) within each cross-validation fold to prevent data leakage.

6.4 Model Family 2: Graph2Vec + ML

The Graph2Vec pipeline combines unsupervised graph embedding with supervised classification:

Step 1: The `WeisfeilerLehmanKernel` class (see Section 2.4.3) extracts structural features from each graph using 3 WL iterations. Each node is relabeled based on its neighborhood structure, and the resulting labels form the "words" of the graph's "document."

Step 2: The `Graph2Vec` class trains a Doc2Vec model (DBOW, 128 dimensions, 100 epochs) on the corpus of graph-documents to produce a 128-dimensional embedding for each graph.

Step 3: These embeddings replace the flat features as input to the same six baseline classifiers. This tests whether graph structure, as captured by WL-based embeddings, provides additional discriminative power.

Note

A critical design decision in this pipeline is that Graph2Vec embeddings are computed **once on the full dataset** before cross-validation splits. While this could introduce a subtle form of information leakage (test graphs contribute to the embedding space), it is consistent with the original Graph2Vec paper's evaluation protocol and follows Stoppa (2023)'s methodology. A stricter approach would re-train Graph2Vec within each fold, which we discuss in the Limitations chapter.

6.5 Model Family 3: GCNN (End-to-End)

The GCNN represents the most sophisticated model family, learning directly from graph structure without any intermediate embedding step.

The `GCNN` class implements a 2-layer graph convolutional network (Equation 2.1) with:

- Input dimension: 6 (node feature vector)
- Hidden dimension: 64
- Dropout: 0.3
- Global mean pooling for graph-level representation
- Task-specific output heads (1 for binary, 3/5 for multiclass, 1 for regression)

Training uses Adam optimizer (learning rate 0.005, weight decay 5×10^{-4}) with early stopping (patience 30 epochs). The `GCNNTrainer` class manages the training loop, including custom `collate_graphs` batching that constructs block-diagonal adjacency matrices for mini-batch processing.

6.6 Evaluation Framework

The `EvaluationFramework` class orchestrates all experiments:

1. Generate 5-fold stratified splits (shared across all models for fair comparison)
2. Evaluate all baseline models on all tasks
3. Compute Graph2Vec embeddings and evaluate with all ML models
4. Train and evaluate GCNN on all tasks
5. Run ablation study (see next section)
6. Perform statistical significance testing between model families
7. Generate comprehensive CSV reports

6.7 Ablation Study

The ablation study removes one feature branch at a time and measures the impact on performance. For each of the seven branches:

1. Remove all columns belonging to that branch from the flat feature matrix

2. Re-train and re-evaluate the Gradient Boosting model (chosen as the representative baseline)
3. Compute $\Delta = \text{acc}_{\text{full}} - \text{acc}_{\text{ablated}}$

A positive Δ indicates that removing the branch **hurts** performance (the branch is important). A negative Δ indicates that removing the branch **improves** performance (the branch may be introducing noise or redundancy).

6.8 Expected Output

After running Part 3:

- `results/model_results.csv`: Complete performance table (42 rows $\times$ 17 metric columns)
- `results/ablation_results.csv`: Ablation study results (16 rows: full + 7 branches $\times$ 2 tasks)
- `results/all_results.pkl`: Serialized Python object with per-fold metrics for deeper analysis

Chapter 7

Pipeline Part 4: Evaluation and Visualization

Code Reference

File: `src/part4_evaluation.py` (1,079 lines)
Run: `python src/part4_evaluation.py`
Input: `results/model_results.csv`, `results/ablation_results.csv`, `results/all_results.pkl`
Output: 10 publication-ready figures + text evaluation report
Dependencies: matplotlib, seaborn, numpy, pandas

7.1 What This Code Does

Part 4 transforms the raw numerical results from Part 3 into publication-quality visualizations and a comprehensive text report. It generates 10 figures (each saved as PNG) and a text-based evaluation report, using 15 specialized functions.

7.2 Generated Figures

1. **fig01_accuracy_heatmap.png**: Heatmap showing accuracy (mean ± std) for all model-task combinations
2. **fig02_model_comparison_bars.png**: Grouped bar chart comparing model families across tasks
3. **fig03_binary_metrics_radar.png**: Radar chart of precision/recall/F1 for binary classification
4. **fig04_multiclass3_confusion.png**: Confusion matrix for best 3-class model
5. **fig05_regression_scatter.png**: Predicted vs. actual MMSE scatter plot with regression line
6. **fig06_ablation_impact.png**: Horizontal bar chart showing ablation Δ for each branch
7. **fig07_family_boxplot.png**: Box plots of per-fold accuracy distributions by model family

8. **fig08_task_difficulty.png**: Line plot showing accuracy degradation from binary → 5-class
9. **fig09_regression_residuals.png**: Residual analysis plot for MMSE regression
10. **fig10_summary_dashboard.png**: 2×2 summary dashboard combining key insights

Note

You can insert the generated figures from `results/figures/` into this document at the appropriate locations. In particular, `fig10_summary_dashboard.png` provides a compact visual overview of the entire project's results and is ideal for presentations.

7.3 Evaluation Report

The text report (`results/evaluation_report.txt`) contains:

- Complete performance tables for all 42 model-task configurations
- Best model identification per task
- Statistical significance test results (paired t-tests between model families)
- Ablation study summary with branch importance rankings
- Key findings and observations

Chapter 8

Results and Analysis

This chapter presents the experimental results across all four prediction tasks, three model families, and the ablation study.

8.1 Overall Model Performance

Table 8.1 shows the best model from each family across all classification tasks.

Table 8.1: Best Model Performance by Family and Task (Accuracy ± Std)

Task	Baseline (SVM)	Graph2Vec (NN)	GCNN
Binary	**82.8% ± 3.3%**	66.7% ± 4.6%	60.9% ± 0.2%
3-Class	**83.2% ± 3.7%**	65.5% ± 4.2%	57.7% ± 2.2%
5-Class	**58.5% ± 3.0%**	46.5% ± 0.5%	45.9% ± 0.5%

Table 8.2: MMSE Regression Performance (Best per Family)

Metric	Baseline (GBT)	Graph2Vec (GBT)	GCNN
RMSE	**4.16 ± 0.32**	5.28 ± 0.31	5.64 ± 0.65
MAE	**3.36 ± 0.29**	4.41 ± 0.29	4.80 ± 0.74
R^2	**0.40 ± 0.06**	0.04 ± 0.07	−0.11 ± 0.21

8.2 Key Finding: Baseline Dominance

The most striking result is the clear superiority of baseline models over graph-based approaches. The SVM operating on flat 348-feature vectors outperforms both Graph2Vec and GCNN by substantial margins across all tasks. This finding is consistent with Stoppa (2023), who observed the same pattern on the English-language DementiaBank corpus.

Why do baselines win? Several factors contribute:

1. **Dataset size**: With only 500 conversations, the dataset is too small for deep graph learning methods to realize their potential. GCNNs typically require thousands of graphs to learn meaningful structural patterns.

2. **Feature quality**: The 348 hand-crafted features are highly informative and already capture the relevant signal. The graph structure adds topological information but does not add new feature content.

3. **Information bottleneck**: When converting 348 features into a 128-dimensional Graph2Vec embedding, significant information is lost. The WL kernel captures structural patterns but may not preserve the discriminative feature values.

4. **Tree uniformity**: Because every graph follows the same ontology tree structure, the structural variation between graphs is minimal. Graph methods excel when graphs have diverse topologies.

8.3 GCNN Majority-Class Collapse

The GCNN's binary accuracy of 60.9% with near-zero variance (±0.2%) is a telltale sign of **majority-class collapse**: the model learns to predict the majority class (Dementia = 60.4% of samples) for virtually all inputs. The F1-score of 0.46 confirms this, a model predicting only the majority class would achieve approximately 0.46 weighted F1.

This behavior is common in GCNNs on small graph datasets and indicates that the model fails to learn discriminative features through the graph convolution layers. The 6-dimensional node features (5 type bits + 1 value) may be too sparse for effective message passing.

8.4 Task Difficulty Gradient

Performance degrades monotonically from binary → 3-class → 5-class, reflecting the increasing difficulty of finer-grained distinctions:

- **Binary** (82.8%): The clearest clinical boundary, separating healthy from impaired
- **3-Class** (83.2%): Interestingly, slightly higher than binary for SVM due to better class separation
- **5-Class** (58.5%): Substantially harder, with the "Uncertain" class being particularly difficult

8.5 MMSE Regression Analysis

The Gradient Boosting baseline achieves RMSE = 4.16, meaning the model's predictions are off by approximately 4 MMSE points on average. With the MMSE range being 8–30

in this dataset, this represents a meaningful but imperfect prediction. The $R^2 = 0.40$ indicates the model explains 40% of the variance in MMSE scores.

Graph2Vec models achieve near-zero R^2 (0.04), indicating they fail to capture the continuous relationship between features and MMSE scores. The GCNN achieves *negative* R^2 (-0.11), performing worse than simply predicting the mean MMSE for all patients.

8.6 Ablation Study: Which Features Matter Most?

The ablation study reveals which feature branches contribute most to classification performance. Table 8.3 shows the impact of removing each branch:

Table 8.3: Ablation Study Results (Binary Classification, Gradient Boosting)

Branch Removed	**Accuracy (%)**	**Δ (%)**	**Interpretation**
Full model (none removed)	81.0	—	Baseline reference
Discourse (35 feat.)	70.9	+10.0	**Most important**—large drop when removed
Psycholinguistic (6 feat.)	79.6	+1.4	Important—modest drop
Acoustic (170 feat.)	80.0	+1.0	Important—small drop
Anagraphic (4 feat.)	80.4	+0.6	Mildly important
Syntactic (55 feat.)	81.4	−0.4	Redundant—slight improvement when removed
Lexical (33 feat.)	81.4	−0.4	Redundant—slight improvement when removed
Spatial (40 feat.)	81.6	−0.6	Redundant—modest improvement when removed

Note

The Δ column uses the convention: positive Δ = performance drops when branch is removed (branch is important); negative Δ = performance improves when branch is removed (branch may be redundant or noisy). A small negative Δ is within noise range and should not be over-interpreted.

Key findings from ablation:

- **Discourse features are overwhelmingly the most important**, with a 10 percentage point drop when removed. This aligns with clinical evidence that discourse organization is a sensitive marker of cognitive decline.
- **Psycholinguistic features** contribute modestly despite having only 6 features, suggesting high information density per feature.
- **Acoustic features** contribute less than expected given their large dimensionality (170 features), indicating potential redundancy among the 14 MFCC channels.

- **Spatial, Lexical, and Syntactic features** appear redundant with other branches, removing them slightly improves performance, possibly because they introduce noise or multicollinearity.

Chapter 9

Development Journey: Debugging and Corrections

Building a research pipeline of this complexity inevitably involves encountering and resolving bugs. This chapter documents the significant issues we discovered during development and how we corrected them. We include this not as an admission of error but as an educational resource, these are precisely the kinds of issues that practitioners encounter in real machine learning projects, and learning to diagnose and fix them is a crucial skill.

9.1 Part 3 Code Review: Six Critical Bugs

Before executing the ML models pipeline, we conducted a thorough code review of Part 3 and identified six critical bugs that would have produced incorrect results or outright crashes.

9.1.1 C1: GCNN Zero-Feature Bug

Problem: The GCNN's `prepare_graph_data` method extracted node features by looking for an `'x'` attribute on graph nodes. However, the graphs produced by Part 2 stored features as `'node_type'`, `'feature_value'`, and `'group_label'` attributes, there was no `'x'` key. The code's fallback was `G.nodes[node].get('x', [0]*6)`, which meant every node received an all-zeros feature vector.

Impact: The GCNN would train on graphs with zero information content, learning nothing and collapsing to majority-class prediction.

Fix: We modified the `GraphManipulator.prepare_for_ml()` method in Part 2 to convert node attributes into proper `'x'` feature vectors (5-dimensional one-hot type encoding + normalized feature value) before passing them to Part 3.

9.1.2 C2: Graph2Vec Data Leakage

Problem: The original code trained Graph2Vec on the *entire* dataset before splitting into cross-validation folds. This means test graphs contributed to the learned embedding space, a form of transductive information leakage.

Impact: Inflated Graph2Vec performance, making graph-based methods appear more competitive than they actually are.

Fix: We acknowledged this as a known limitation consistent with the original Graph2Vec evaluation protocol. The fixed code documents this clearly and reports results with appropriate caveats. A stricter approach would retrain Graph2Vec within each fold.

9.1.3 C3: Memory Overflow from Dense Batching

Problem: The `collate_graphs` function constructed dense block-diagonal adjacency matrices for mini-batches. With 455 nodes per graph and batch size 32, this creates a $14{,}560 \times 14{,}560$ dense matrix per batch, requiring approximately 800 MB of RAM.

Impact: Out-of-memory crashes on systems with limited RAM.

Fix: Reduced batch size to 32 (from the original 64) and added proper memory management. An alternative solution using sparse matrices was documented for future implementation.

9.1.4 C4: Variable Scoping in Early Stopping

Problem: The GCNN training loop checked `if 'best_state' in dir()` to determine whether a best model checkpoint existed. However, `dir()` in Python returns names in the local scope, and `best_state` from a previous fold could persist, causing the current fold to load stale weights.

Impact: Wrong model weights loaded for evaluation, producing unreliable per-fold metrics.

Fix: Initialize `best_state = None` before each fold and check `if best_state is not None`.

9.1.5 C5: Missing XGBoost Dependency

Problem: The code imported `xgboost` at the top of the file but never actually used it in any model. This would cause an ImportError on systems without XGBoost installed.

Impact: Script crash before any computation.

Fix: Removed the unused import.

9.1.6 C6: Sample Misalignment

Problem: The flat features file contained 500 rows while the graph container had only 499 entries (due to a duplicate conversation ID `IND-0159-extra`). When the ablation study used both flat features and graph-aligned labels, the sample indices would not match.

Impact: Training on misaligned features and labels, producing meaningless results.

Fix: Added explicit alignment checks and deduplication logic to ensure flat features and graph labels have exactly the same samples in the same order.

9.2 Part 4 Windows Deployment: Four Additional Bugs

When deploying the pipeline on Windows systems, four additional issues emerged:

1. **NumPy 2.x / PyTorch 1.x incompatibility**: The `torch.randperm().numpy()` call failed because NumPy 2.x changed its C API. Fixed by pinning `numpy<2` in requirements.
2. **Multiprocessing failures**: `BaggingClassifier` and `BaggingRegressor` with `n_jobs=-1` caused pickling errors on Windows due to its lack of fork-based multiprocessing. Fixed by setting `n_jobs=1`.
3. **Unicode encoding errors**: The evaluation report used Greek delta characters (Δ) which failed with the default Windows `cp1252` encoding. Fixed by explicitly specifying `encoding='utf-8'` in all file write operations.
4. **GCNN RuntimeError**: Related to the NumPy 2.x issue—`torch.randperm(n).numpy()` triggered a compatibility error. Same fix as Bug 1.

Note

Documenting these debugging journeys serves an educational purpose. In real-world ML projects, a significant portion of development time is spent diagnosing and fixing exactly these kinds of issues,data misalignment, cross-platform compatibility, dependency conflicts, and subtle statistical errors. The ability to systematically identify root causes and implement targeted fixes is a hallmark of mature engineering practice.

Chapter 10

Limitations and Future Directions

10.1 Current Limitations

10.1.1 Dataset Constraints

- **Sample size**: With 500 conversations, the dataset is at the lower end for training deep learning models. GCNNs and Graph2Vec both benefit from larger training sets, and the current results may underestimate their potential.
- **Class imbalance**: The 60/40 dementia-to-control split, while clinically realistic, presents challenges for minority-class prediction. The 5-class task is particularly affected, with some severity levels having very few samples.
- **Single assessment type**: All conversations follow the same picture description protocol. Real-world deployment would benefit from multi-task assessments.

10.1.2 Methodological Limitations

- **Graph2Vec leakage**: As discussed in Section 6.3, Graph2Vec embeddings are computed on the full dataset before cross-validation. A stricter evaluation would retrain Graph2Vec within each fold.
- **Uniform graph topology**: All conversation graphs share the same tree structure (same ontology). This limits the ability of graph methods to exploit topological variation,they can only differentiate graphs by node feature values, not by structure.
- **GCNN architecture**: The 2-layer GCN with 6-dimensional input features may be too shallow and narrow for the deep tree structure (depth 5). More expressive architectures (GAT, GraphSAGE, deeper networks) might perform better.
- **Cross-validation limitations**: With only 5 folds, the variance of performance estimates is inherently high. Results should be interpreted with appropriate uncertainty.

10.1.3 Clinical Limitations

- **External validation**: The model has not been validated on an independent external dataset from different clinical sites.
- **Longitudinal tracking**: The current approach evaluates single snapshots. Tracking changes over time could improve early detection sensitivity.
- **Indonesian language specificity**: While the feature engineering framework is language-agnostic in principle, the specific feature generators would need adaptation for other Indonesian dialects or languages.

10.2 Future Directions

10.2.1 Immediate Extensions

1. **Richer graph architectures**: Replace the fixed ontology tree with learned graph structures, or use heterogeneous graph neural networks that can model different edge types (branch-category, category-feature, cross-branch correlations).
2. **Attention-based models**: Graph Attention Networks (GATs) (Veličković et al., 2018) could learn which features and branches to attend to, potentially improving both performance and interpretability.
3. **Cross-branch connections**: The current tree structure has no edges between different branches. Adding learned cross-branch connections (e.g., acoustic $\leftrightarrow$ discourse) could capture inter-domain correlations.
4. **Feature selection**: The ablation study suggests that some branches (Spatial, Lexical) may be redundant. Systematic feature selection could improve performance by reducing noise.

10.2.2 Longer-Term Research

1. **Multilingual extension**: Adapt the framework for other Southeast Asian languages (Thai, Vietnamese, Malay) to enable cross-linguistic studies of dementia markers.
2. **Multimodal integration**: Incorporate video-based features (facial expressions, gesture analysis) alongside speech features for richer clinical assessment.
3. **Transformer-based approaches**: Apply pre-trained language models (e.g., IndoBERT for Indonesian) for feature extraction, potentially replacing handcrafted features with learned representations.
4. **Federated learning**: Enable multi-site collaborative model training without sharing patient data, addressing privacy concerns in clinical settings.

5. **Clinical deployment**: Develop a user-friendly interface for clinicians, including uncertainty quantification and explainability features that highlight which speech characteristics drove the model's prediction.

6. **Longitudinal graph sequences**: Model disease progression by constructing temporal sequences of conversation graphs and using graph-level sequence models (e.g., graph-level LSTMs) to predict cognitive trajectory.

10.3 Reproducibility Checklist

To reproduce all results in this document:

1. Install dependencies: `pip install -r requirements.txt`
2. Run Part 1: `python src/part1_dataset_generation.py`
3. Run Part 2: `python src/part2_graph_construction.py`
4. Run Part 3: `python src/part3_ml_models.py`
5. Run Part 4: `python src/part4_evaluation.py`

All scripts use `random_seed=42` for reproducibility. Expected total runtime on a modern laptop (no GPU required) is approximately 15–30 minutes.

Bibliography

Alzheimer's Indonesia (2019). *Dementia in Indonesia: Statistics and Challenges*. Alzheimer's Indonesia, Jakarta.

Becker, J. T., Boller, F., Lopez, O. L., Saxton, J., and McGuff, K. L. (1994). The natural history of Alzheimer's disease: Description of study cohort and accuracy of diagnosis. *Archives of Neurology*, 51(6):585–594.

Bucks, R. S., Singh, S., Cuerden, J. M., and Wilcock, G. K. (2000). Analysis of spontaneous, conversational speech in dementia of Alzheimer type: Evaluation of an objective technique for analysing lexical performance. *Aphasiology*, 14(1):71–91.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 2nd edition.

Farzana, S. and Parde, N. (2020). Exploring MMSE score prediction using verbal and non-verbal cues. In *Proceedings of INTERSPEECH 2020*, pages 2207–2211.

Folstein, M. F., Folstein, S. E., and McHugh, P. R. (1975). "Mini-mental state": A practical method for grading the cognitive state of patients for the clinician. *Journal of Psychiatric Research*, 12(3):189–198.

Fraser, K. C., Meltzer, J. A., and Rudzicz, F. (2015). Linguistic features identify Alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422.

Joty, S., Carenini, G., and Ng, R. T. (2015). CODRA: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1188–1196.

Lu, X. (2010). Automatic analysis of syntactic complexity in second language writing. *International Journal of Corpus Linguistics*, 15(4):474–496.

Morris, J. C. (1993). The Clinical Dementia Rating (CDR): Current version and scoring rules. *Neurology*, 43(11):2412–2414.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.

Ng, K. P., Chiew, H. J., Rosa-Neto, P., Bhatt, D. L., and Bhatt, P. P. (2018). The influence of language and culture on cognitive assessment tools in the diagnosis of early cognitive impairment and dementia. *Expert Review of Neurotherapeutics*, 18(11):859–869.

Perneczky, R., Wagenpfeil, S., Komossa, K., Grimmer, T., Diehl, J., and Kurz, A. (2006). Mapping scores onto stages: Mini-mental state examination and clinical dementia rating. *American Journal of Geriatric Psychiatry*, 14(2):139–144.

Sahidullah, M. and Saha, G. (2012). Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. *Speech Communication*, 54(4):543–565.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561.

Stoppa, E. (2023). *GraDD: A Graph Machine Learning and Natural Language Processing Approach to Automatic Dementia Detection*. Master's Thesis, University of Illinois at Chicago.

Stoppa, E., Di Donato, G. W., Parde, N., and Santambrogio, M. D. (2022). Computer-aided dementia detection: How informative are your features? In *IEEE International Forum on Research and Technologies for Society and Industry (RTSI)*, pages 55–61.

Suryadinata, R. V., Wirjatmadi, B., and Adriani, M. (2021). Cognitive impairment among Indonesian elderly: Prevalence and risk factors. *Journal of Public Health Research*, 10(2).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.