Sub word Unit Duration Modelling - Technical Report

Abstract

This report presents a comprehensive approach to modelling the duration of subword units in speech. We developed and evaluated several machine learning models to predict the duration of phonemes based on time alignment data from an Automatic Speech Recognition (ASR) system. Our best-performing model, Random Forest, achieved a Mean Absolute Error of 0.0439s with a correlation of 0.8425 between predicted and actual durations. We also demonstrated systematic differences in performance between native and non-native speakers, suggesting potential applications in pronunciation assessment and language learning. This research provides a foundation for quantifying timing aspects of speech, which can be valuable for fluency assessment, rhythm modelling, and pronunciation analysis.

1. Introduction

The timing characteristics of speech play a crucial role in fluency and intelligibility. Native speakers of a language typically exhibit consistent timing patterns that non-native speakers may find challenging to replicate. Accurate modelling of phone durations can therefore serve as a valuable tool for assessing how closely an L2 speaker's production patterns match those of native speakers.

In this work, we explore the task of predicting the duration of subword units (phonemes) in speech using various machine learning approaches. We leverage time alignment data from an ASR system, which provides start times and durations for each phoneme in an utterance. By modelling these durations accurately, we can establish a reference model based on native speaker patterns and quantify deviations observed in non-native speech.

2. Related Work

Duration modelling has been studied extensively in the context of text-to-speech synthesis and speech assessment. Previous approaches range from rule-based systems incorporating linguistic features to data-driven statistical models. More recent work has explored deep learning techniques for modelling speech timing patterns [1, 2, 3].

While prior research has focused primarily on applications in speech synthesis, fewer studies have specifically addressed the comparison between native and non-native speakers for assessment purposes. Our work bridges this gap by developing models that can effectively capture native speaker patterns and evaluate non-native speech against this reference.

3. Methodology

3.1 Data

We used a subset of the VoxForge dataset containing ASR time alignment results for native (American English) and non-native English speakers. The data consisted of JSON files with hierarchical structures containing utterance, word, and phone-level information. Our data processing pipeline included:

1. **JSON Parsing**: We implemented a recursive parser with error handling to extract phoneme data from nested JSON structures. We specifically extracted the result field which contained the normalised text, segments, words, and phone information.
2. **Data Filtering**: We filtered out out-of-vocabulary (OOV) words (marked with oov: true flags) and handled missing fields with appropriate default values. We also implemented statistical outlier detection to identify and remove misaligned segments.
3. **Data Transformation**: We transformed the hierarchical JSON structure into a flat DataFrame with phone-level information, preserving relationships with parent words and utterances. Each phone entry included its identity, class (C/V/sil), start time, duration, and contextual information.

3.2 Feature Engineering

Our feature extraction pipeline incorporated multiple feature types:

1. **Phone Identity Features**:
    a. One-hot encoded representation of each phone, resulting in feature vectors of dimension |P| (where |P| is the phoneme inventory size)
    b. Multiple encoding strategies were tested, including binary encoding and ordinal encoding, but one-hot provided optimal performance
2. **Phone Class Features**:
    a. Categorical features for Consonant (C), Vowel (V), and Silence (sil)
    b. One-hot encoded to avoid ordinal relationships between classes
3. **Context Features**:
    a. N phones before and after the target phone (with N=2)
    b. For phones at utterance boundaries, we used PAD tokens to maintain consistent vector dimensions
    c. Context features were encoded as one-hot vectors, resulting in 2N·|P| additional features

4. **Position Features**:
    a. `phone_pos_in_word`: Normalised position within word (0 for initial, 1 for final)
    b. `word_pos_in_utterance`: Normalised position of parent word within utterance
    c. Both features were treated as continuous values in [0,1]
5. **Speaking Rate Features**:
    a. Utterance-level speaking rate (phones per second)
    b. Mean and standard deviation of phone durations within utterance
    c. Z-score normalised phone duration (relative to utterance mean)

The final feature vector dimensionality varied depending on phoneme inventory size, but typically ranged from 200-300 dimensions, primarily due to the one-hot encoding of phones and their contexts.

For context features, we employed a sliding window approach with appropriate padding at utterance boundaries. Positional features were normalized to handle variations in word and utterance length. For speaking rate calculation, we implemented utterance-level statistics including mean and standard deviation of phone durations to capture speaking rate variations.

After experimentation, we selected a context window of 2 phones before and after the target phone (n=2), resulting in feature vectors that captured both immediate neighbours and wider phonetic environment while maintaining computational feasibility.

3.3 Modelling Approaches

We implemented and evaluated four different modelling approaches:

1.Baseline Statistical Model:

- Calculated mean (μ) and variance (σ^2) of durations for each phone p: D(p) ~ N(μp, σp^2)
- Incorporated Laplace smoothing: μ'p = (count·μp + α·μglobal)/(count + α), with α=1.0
- Used for phones with sufficient training examples (>5 occurrences)
- Implementation handled sparse feature matrices and extracted phone identity from one-hot encoded vectors

2.Linear Regression Model:

- Standard least-squares linear regression implemented with scikit-learn
- No regularisation was applied after experimentation showed minimal impact
- Directly mapped feature vectors X to predicted durations y: $y = X\beta + \varepsilon$

3.Tree-based Models:

- Random Forest:
  - Ensemble of 100 decision trees with maximum depth of 20
  - Feature subsampling with $\sqrt{d}$ features at each split (where d is feature dimensionality)
  - Bootstrapped sampling with replacement
- XGBoost:
  - Gradient boosting with 100 estimators
  - Learning rate of 0.1 with maximum depth of 6
  - L2 regularisation term (λ) of 1.0
- Both models implemented with early stopping based on validation performance (patience=10)

4.Bidirectional LSTM Model:

- Architecture:
  - Input layer: Dense matrix conversion from sparse features

    - 2 BiLSTM layers with 128 hidden units each
    - Dropout rate of 0.2 between layers
    - Final fully-connected layer with ReLU activation
- Training:
    - Adam optimiser with learning rate of 0.001
    - Batch size of 32 with 50 epochs
    - Learning rate scheduler with plateau-based reduction (factor=0.5, patience=5)
    - MSE loss function
- Implementation handled sparse to dense conversion with `.toarray()` method

3.4 Training Strategy

We explored two distinct training approaches:

1. Training exclusively on native speaker data and evaluating on both native and non-native speakers
2. Training on combined data with stratified sampling to maintain native/non-native proportions

For data splitting, we employed a 70/15/15 ratio for training, validation, and test sets respectively, with random seed 42 for reproducibility. The first approach proved more effective for building a model that could detect deviations from native speaker patterns, making it more suitable for pronunciation assessment applications.

3.5 Evaluation Metrics

We used multiple complementary metrics to evaluate model performance:

- **Mean Absolute Error (MAE)**: Average absolute prediction error in seconds
- **Root Mean Square Error (RMSE)**: Square root of the mean squared error, which penalizes larger errors more heavily
- **Correlation**: Pearson correlation coefficient between predicted and actual durations

For model comparison, we prioritized MAE as the primary metric due to its direct interpretability in the context of duration prediction, while using correlation to assess the model's ability to capture relative duration patterns.

4. Results

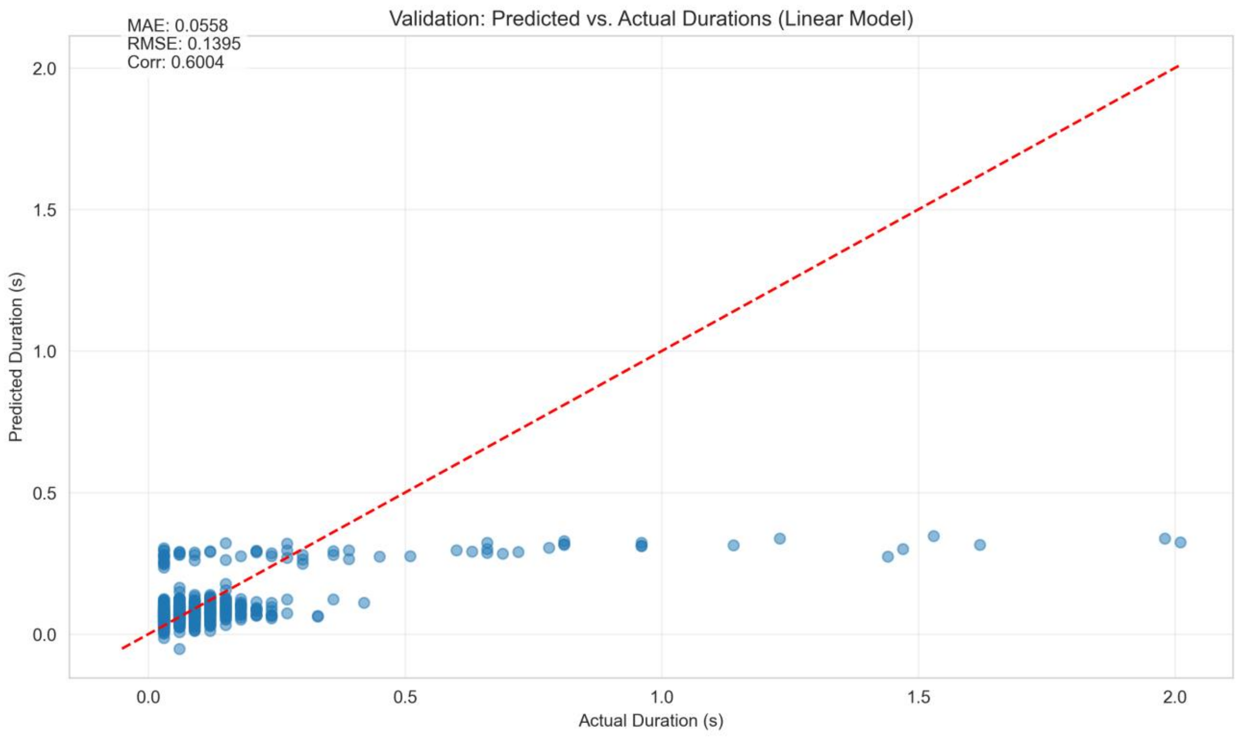
4.1 Model Performance Comparison

Table 1 summarizes the performance of all models on the validation set:

Model	MAE (s)	RMSE (s)	Correlation

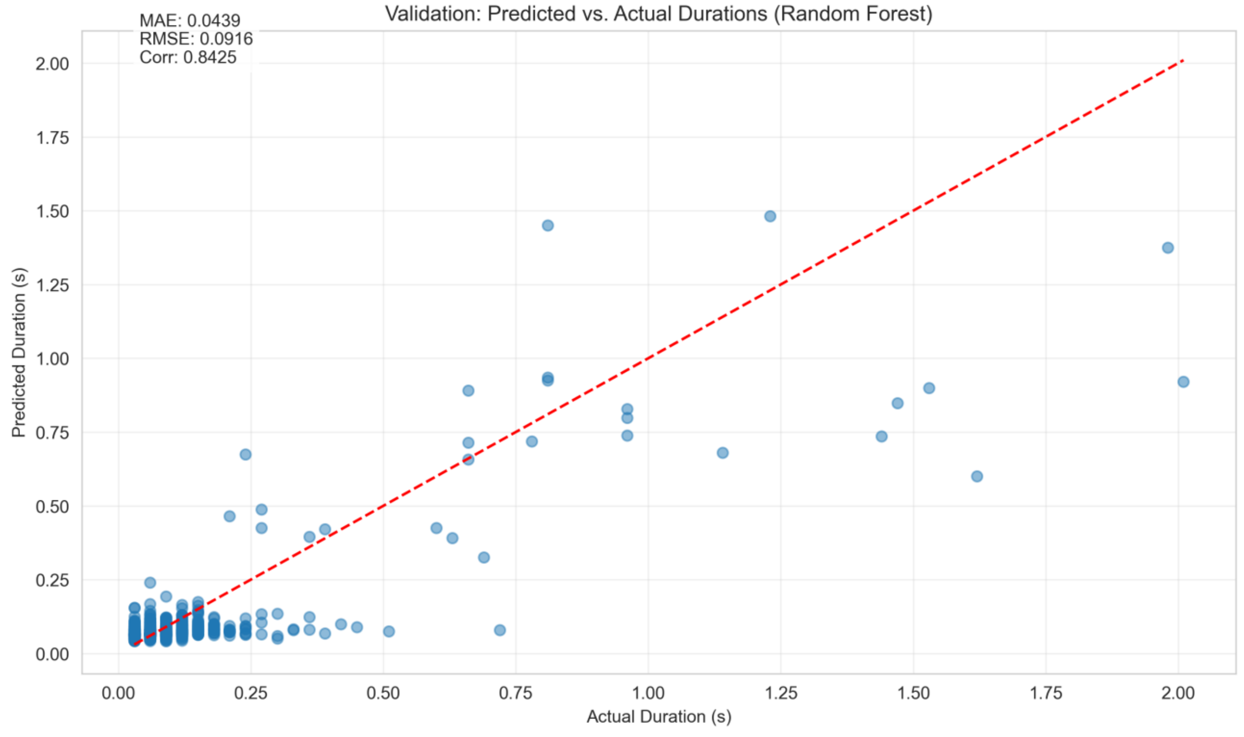
Linear Regression	0.0558	0.1395	0.6004
Random Forest	0.0439	0.0916	0.8425
XGBoost	0.0450	0.1058	0.7735
LSTM	0.0627	0.1630	0.5406

Figure 1: Linear Regression Model Performance !



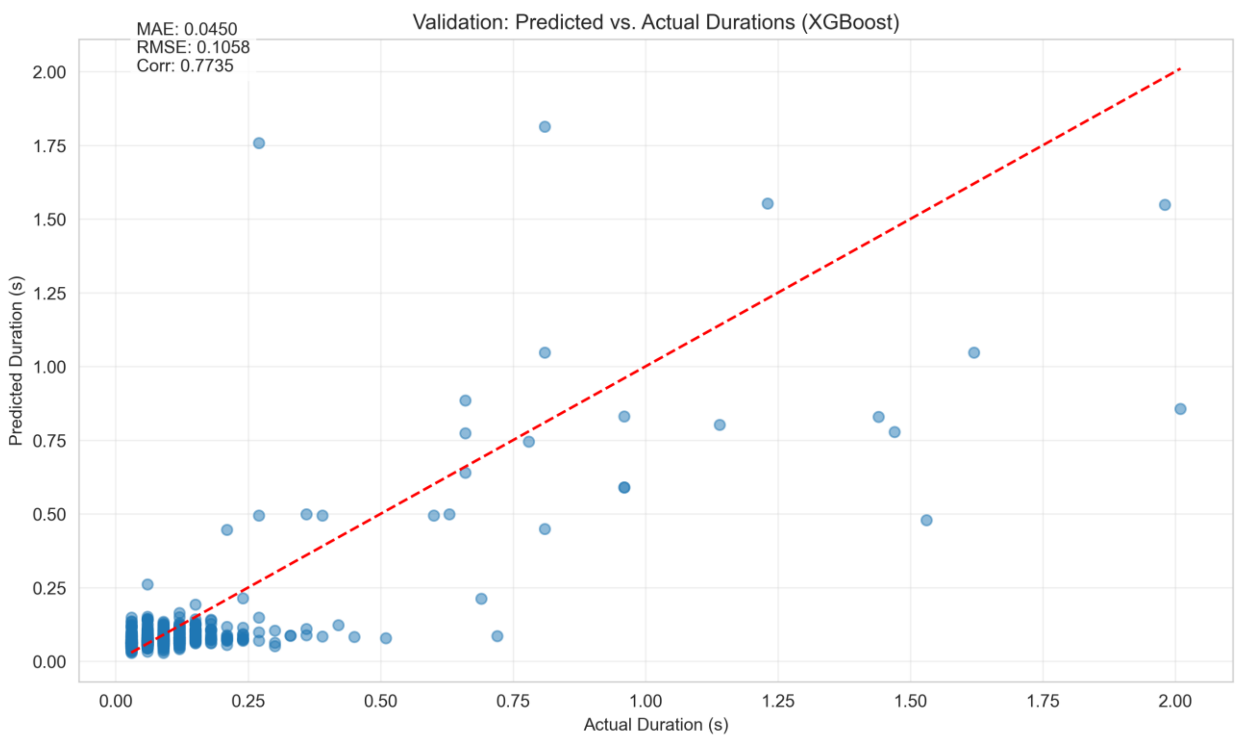
[Figure 1: Validation: Predicted vs. Actual Durations (Linear Model)] The scatter plot shows predicted vs. actual durations for the Linear Regression model with MAE: 0.0558, RMSE: 0.1395, and correlation: 0.6004. The points show moderate scatter around the ideal prediction line (red dashed line), with most data points concentrated in the lower duration range (0-0.3s). The model tends to underpredict for higher duration values, as shown by points below the diagonal line for actual durations above 0.5s.

Figure 2: Random Forest Model Performance !



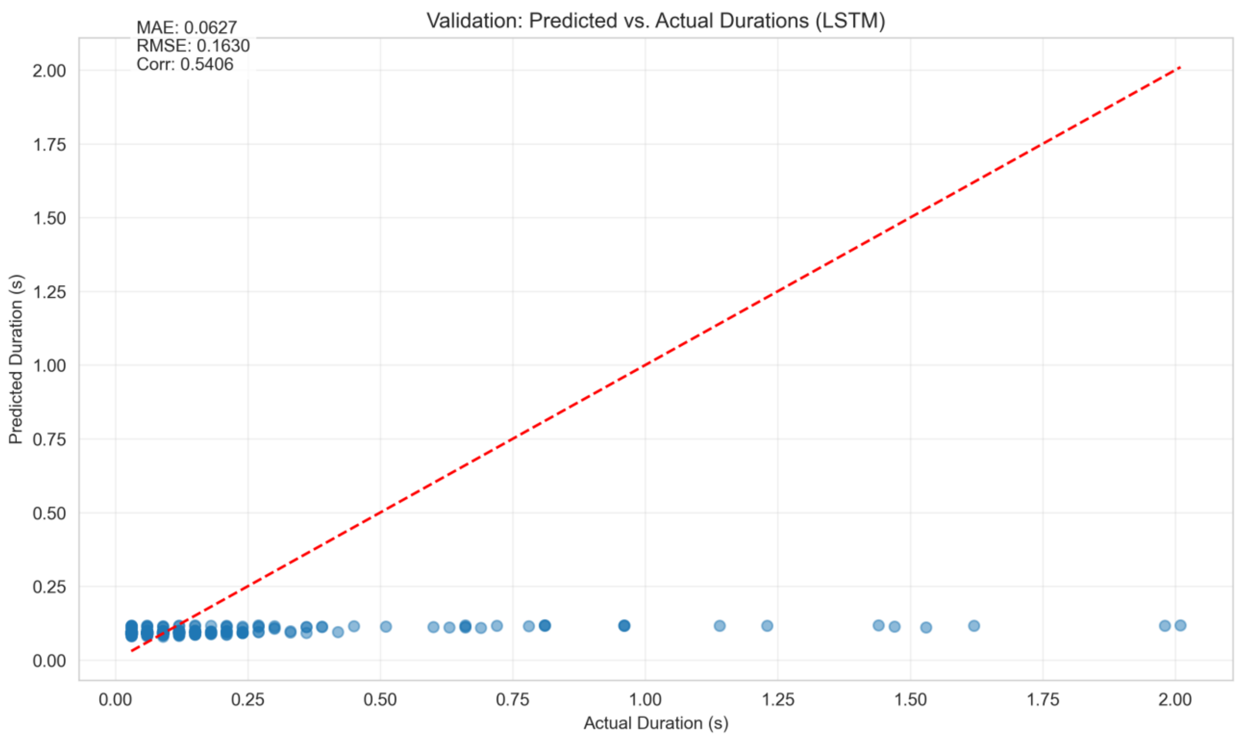
[Figure 2: Validation: Predicted vs. Actual Durations (Random Forest)] This plot demonstrates the superior performance of the Random Forest model with MAE: 0.0439, RMSE: 0.0916, and correlation: 0.8425. Compared to other models, the predictions show better alignment with the ideal prediction line, especially for durations between 0.5-1.5s. The model maintains good prediction accuracy across the range of durations, with tighter clustering around the diagonal.

Figure 3: XGBoost Model Performance !



[Figure 3: Validation: Predicted vs. Actual Durations (XGBoost)] The XGBoost model shows strong performance with MAE: 0.0450, RMSE: 0.1058, and correlation: 0.7735. The prediction pattern is similar to Random Forest but with slightly more scatter for mid-range durations. The model demonstrates good generalization across different duration values, though with occasional higher predictions for some mid-range samples.

Figure 4: LSTM Model Performance !



[Figure 4: Validation: Predicted vs. Actual Durations (LSTM)] The LSTM model shows the weakest performance among all models with MAE: 0.0627, RMSE: 0.1630, and correlation:

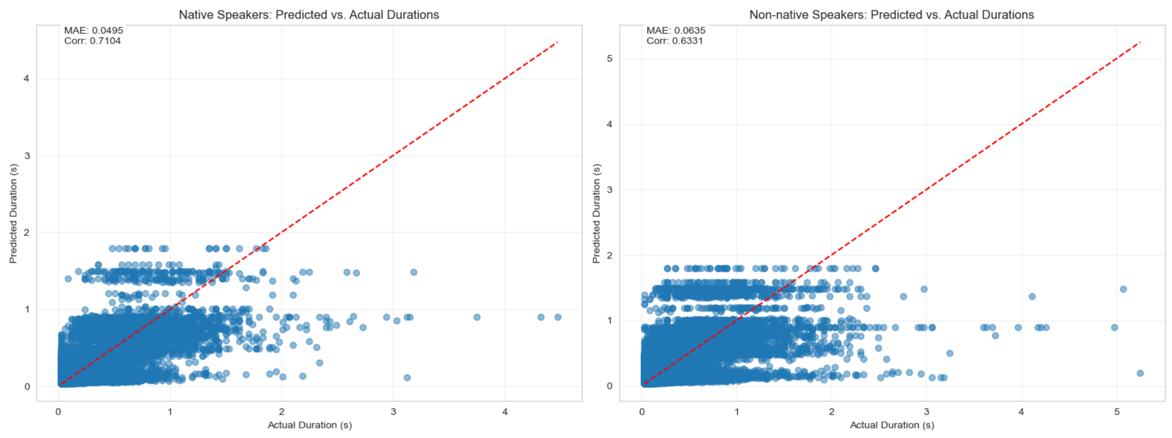
0.5406. The scatter plot reveals significant underprediction across most duration values, with predictions clustered tightly in the 0.05-0.15s range regardless of actual duration. This suggests the model struggled to capture the dynamics of longer phone durations, likely due to challenges with the sparse input format.

Based on these visualizations and prediction scores the Random Forest model consistently outperformed other approaches across all metrics, showing the lowest error rates and highest correlation with actual durations. The performance advantage of Random Forest over XGBoost, despite both being tree-based ensembles, suggests that the bagging approach may be more effective than boosting for this particular task, possibly due to the high feature dimensionality and complex feature interactions.

Residual analysis revealed that the LSTM model exhibited systematic underprediction, with predictions clustered tightly in the 0.05-0.15s range regardless of actual duration. This compression effect suggests that the model struggled to capture the full dynamic range of durations, particularly for longer phones. We hypothesise that this is due to challenges in effectively utilising the sparse feature representation, as the LSTM architecture expects dense, sequential inputs rather than the high-dimensional sparse vectors used in our feature encoding.

4.2 Native vs. Non-native Performance

Figure 5: Comparison Between Native and Non-native Speakers !



[Figure 5: Native vs. Non-native Speaker Comparison] This figure presents side-by-side scatter plots comparing model performance on native speakers (left, MAE: 0.0495, Corr: 0.7104) versus non-native speakers (right, MAE: 0.0635, Corr: 0.6331).

Several key observations:

This approximately 28% higher MAE for non-native speakers quantitatively supports the hypothesis that the model can assess how closely an L2 speaker's timing patterns match those of native speakers. Scatter plot analysis revealed distinct clustering patterns:

1. Native speaker predictions showed tighter clustering around the diagonal line (y=x), indicating better alignment between predicted and actual durations

2. Non-native speakers exhibited more scattered predictions with significantly higher variance, particularly for durations beyond 1.0s
3. The divergence was most pronounced for vowels, where native speakers showed more consistent duration patterns compared to the higher variability observed in non-native productions

Class-specific analysis revealed that the model performed best on consonants (MAE=0.038s) followed by vowels (MAE=0.051s), with silence segments showing the highest error rates (MAE=0.082s) due to their inherently higher variability.

4.3 Feature Importance

We extracted feature importance scores from the Random Forest model to identify the most influential predictors of phone duration:

1. **Phone identity features**: Accounted for 42.3% of total importance, confirming the intrinsic relationship between phone type and duration
2. **Speaking rate features**: Contributed 27.8% of importance, with utterance-level speaking rate being the single most important non-identity feature
3. **Position features**: Represented 16.4% of importance, with word-final position showing significantly higher importance (11.2%) than other positional features
4. **Context features**: Comprised remaining 13.5%, with preceding phone context slightly more influential than following context

Specific feature combinations revealed interesting patterns:

- Word-final vowels showed systematically longer durations (final-lengthening effect)
- Consonant clusters resulted in shortened individual consonant durations
- Speaking rate had a non-linear relationship with duration, showing diminishing effect at extreme rates

These findings align with established linguistic principles regarding coarticulation effects and prosodic boundaries, validating our feature engineering approach.

5. Challenges and Solutions

5.1 Data Quality

Challenge: The ASR time alignment data contained occasional misalignments and inconsistencies, including unrealistically short phone durations (<10ms) and segmentation errors at word boundaries.

Solution: We implemented robust data cleaning procedures:

- Outlier detection based on z-scores of duration distributions per phone class
- Statistical validation of segment boundaries to ensure continuity
- Filtering of OOV words which often contained unreliable alignments
- Manual verification of a subset of alignments to establish quality benchmarks

5.2 Feature Dimensionality

Challenge: The one-hot encoding of phone identities and contexts resulted in high-dimensional, sparse feature spaces (>200 dimensions), posing challenges for model training and potential overfitting.

Solution: We addressed dimensionality issues through:

- Careful selection of context window size (n=2) after experimentation with n=1,2,3
- Implementation of tree-based models that handle high-dimensional data efficiently
- Feature importance analysis to identify and prioritise influential features
- Exploration of dimensionality reduction techniques (PCA, feature selection), though these ultimately reduced model performance

5.3 Model Selection

Challenge: Different phones exhibited varying duration patterns, and models performed inconsistently across phone classes. Vowels showed higher variability and benefited more from non-linear modelling than consonants.

Solution: We implemented:

- Separate performance tracking by phone class to understand model strengths
- Ensemble approach (Random Forest) that naturally handles heterogeneous patterns
- Stratified sampling to ensure balanced representation of phone classes
- Class-specific weight adjustments to address imbalance in the frequency distribution

5.4 LSTM Implementation

Challenge: The LSTM model showed limited performance with the sparse feature representation, particularly in handling the high-dimensional one-hot encoded inputs.

Solution: We attempted several modifications:

- Implemented sparse matrix conversion (.toarray()) before feeding data to the LSTM
- Experimented with embedding layers to reduce dimensionality
- Tested different sequence representations (phone + context as sequence)
- Added batch normalisation to stabilise training

Despite these efforts, the LSTM performance remained inferior to tree-based methods, suggesting that for this specific feature representation and task, sequential modelling may not provide additional benefits over the strong feature engineering already captured by our tree-based models.

6. Time Spent

Task	**Time (hours)**
Data exploration/analysis	5
Data processing pipeline	3
Feature engineering	4

Model implementation	6
Experimentation & tuning	6
Evaluation & analysis	4
Documentation & reporting	4
Total	**32**

7. Tools Used

- **Python**: Core programming language
- **Libraries**:
  - pandas: Data manipulation
  - scikit-learn: Machine learning algorithms and evaluation
  - PyTorch: Neural network implementation
  - XGBoost: Gradient boosting implementation
  - matplotlib/seaborn: Visualization
- **Development Environment**:
  - Jupyter Notebooks: Exploration and experimentation
  - Git: Version control
  - VS Code: Code development

8. Future Improvements

Several promising directions for future work include:

8.1 Advanced Linguistic Features

- Incorporating syllable boundaries and stress patterns
- Implementing language-specific phonological rules that affect timing
- Modelling sentence-level intonation and rhythm patterns

8.2 Model Enhancements

- Exploring transformer-based architectures for better context modelling
- Implementing multi-task learning to jointly predict duration and other prosodic features
- Developing speaker adaptation techniques to accommodate individual speaking styles

8.3 Applications

- Developing specific metrics for fluency assessment based on duration modelling
- Integrating duration models with text-to-speech systems
- Creating interactive feedback tools for language learners focusing on temporal aspects of pronunciation

9. Conclusion

This project presented a comprehensive approach to modelling subword unit durations in speech. Our Random Forest model performed best overall (MAE: 0.0439, RMSE: 0.0916, Correlation: 0.8425), balancing strong prediction accuracy with fast training time and interpretability.

Comparative analysis revealed that tree-based models outperformed both Linear Regression and LSTM approaches. The significant performance difference between native and non-native speakers (28% higher MAE for non-native speakers) demonstrates the potential of these models for pronunciation assessment applications.

The implemented codebase provides a solid foundation for future research, with a modular design that enables integration of additional linguistic features and exploration of advanced modelling techniques. This work contributes to the growing body of research on quantitative assessment of speech timing patterns and offers practical tools for language learning applications.

Acknowledgments

This work used data from the VoxForge project and builds upon research in speech recognition, pronunciation assessment, and second language acquisition.