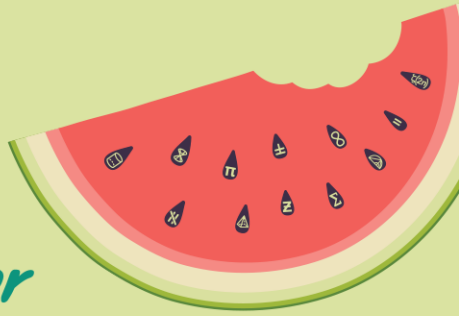
AMSI SUMMER RESEARCH SCHOLARSHIPS 2024–25

Get a taste for Research this Summer



Phonetic Spelling Correction Using Dimensionality Reduction

Louisa Best

Supervised by

Dr. Simon James & Dr. Julien Ugon

Deakin University

February 26, 2025

Jane Street®

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

Abstract

Phonetic spelling correction is essential for individuals who rely on non-standard phoneme-based spellings, including those with intellectual disabilities, early learners, and second-language users. Traditional text-based spell-checking struggles to handle these variations due to the complexity of phonetic similarity. This study investigates the use of dimensionality reduction techniques, specifically Principal Component Analysis (PCA) and Singular Value Decomposition (SVD), to improve phonetic word retrieval by reducing sparse, high-dimensional feature spaces. Both pre-scaled and post-scaled SVD were evaluated in terms of variance retention and phonetic similarity matching. Results indicate that post-scaled SVD significantly enhances phoneme representation, leading to more accurate word predictions when combined with cosine similarity. In contrast, Euclidean and Manhattan distances failed to adequately capture phonetic structure, highlighting the importance of feature scaling in phonetic vector spaces. These findings provide a foundation for further advancements in hybrid models, alternative similarity measures, and context-aware phonetic embeddings, with potential applications in assistive spelling tools, speech recognition, and language learning.

1 Introduction

1.1 Problem Statement and Motivation

Written language presents a barrier for individuals with intellectual disabilities (ID), early learners, and second-language speakers, many of whom rely on phonetic spelling - writing words as they sound rather than following conventional spelling rules. Standard spell checkers, designed for typographical errors, do not address phonetic misspellings, creating a gap in accessible assistive technologies (Garay-Vitoria and Abascal 2006; Dessemontet et al. 2021; Leutzinger 2022). For instance, using “halow” in place of “hello” and “ingre” instead of “injury” are phonetic spellings that standard spell checkers do not identify, underscoring the need for more comprehensive solutions.

Given the orthographic complexity of English, where phonemes (speech sounds) do not map directly to graphemes (letters), individuals using phonetic spelling require specialised correction tools. Existing phonetic algorithms such as *Soundex* and *Double Metaphone* fail in non-standard linguistic contexts, particularly for those with ID (Trinh et al. 2012; Pan, Rickard, and Bjork 2021; Phillips 2000; Knuth 1998).

To address these challenges, this research investigates phonetic spelling errors, existing correction methods, and phoneme feature representation before proposing an approach that takes advantage of n-gram vectorisation and dimensionality reduction to improve phonetic word retrieval. The following sections 1.2-1.5 outline the characteristics of phonetic misspellings, current limitations in phonetic correction, and the research contributions of this study.

1.2 Phonetic Spelling and Types of Spelling Errors

Phonetic misspellings arise due to deviations from conventional orthographic norms. These errors fall into three primary categories:

- **Phonological Errors**: Incorrect or missing phonemes *Example:* "ingre" → "injury" (missing /ʊ/ in /ɪnˈdʒʊri/), or "hello" (Frith 1980).
- **Orthographic Errors**: Incorrect letter substitutions, *Example:* "helo" instead of "hello" or "hallow" instead of "halo" → /həˈloʊ/ (Tong et al. 2009).
- **Morphological Errors**: Incorrect word formation, *Example:* "runned" instead of "ran", "mouses" instead of "mice" (Bahr, Lebby, and Wilkinson 2020).

To effectively tackle these challenges, a reliable phonetic spelling correction system is crucial, as it needs to comprehend the fundamental phoneme representations and precisely link them to the appropriate words.

1.3 Phonetic Spelling Challenges and Existing Approaches

Phonetic spelling, in which words are written based on sound rather than conventional rules, is common among individuals with intellectual disabilities (ID) (Dessemontet et al. 2021; Leutzinger 2022), ESL learners (Nagata, Takamura, and Neubig 2017), and children developing literacy (Brown and Loosemore 1994). Existing algorithms fail to address phonetic misspellings, instead focussing on typographical errors.

Traditional algorithms like *Soundex, Metaphone, and NYSIIS* encode words into phonetic representations but rely on strict letter-sequence rules, making them ineffective for non-standard phonetic variations (Vykhovanets, Du, and Sakulin 2020; Philips 2000). This requires more flexible and data-driven approaches.

1.4 Phonetic Feature Representation and Vectorisation

Phonetic spelling requires alternative linguistic representation beyond rule-based algorithms. Techniques include character-level embeddings, which represent words at the character level and improve the handling of spelling variants (Rubehn et al. 2024); *n*-gram frequency matrices, which capture phoneme cooccurrence (Ryskina 2022; Flint et al. 2017); and word embeddings (Word2Vec, GloVe), which encode words based on semantic similarity but fail to capture phonetic relationships (Mikolov et al. 2013; Pennington, Socher, and Manning 2014). High-dimensional representations pose computational challenges, making dimensionality reduction necessary.

1.5 Research Gap and Project Contribution

Existing models fail in three important aspects: (1) capturing nuanced phoneme-to-grapheme mappings in non-standard spellings (Sofroniev and Çöltekin 2018), (2) efficiently handling high-dimensional phoneme data without losing important structure (Treistman et al. 2022), and (3) generalising well across different spelling patterns in small datasets (Zouhar et al. 2023).

To address these limitations, this study proposes a dimensionality reduction-driven approach to phonetic spelling correction. The method takes advantage of phonetic *n*-gram vectorisation instead of rule-based encodings, applies Singular Value Decomposition (SVD) and Principal Component (PCA) to extract principal

phoneme features while preserving phonetic integrity, and integrates phonetic similarity metrics to improve word retrieval.

The following sections outline the methodology (Section 2), discuss experimental results (Section 3) and summarise findings and explore future directions for improving phonetic spelling models (Section 4).

2 Methodology

Phonetic spelling correction requires a structured data flow to process input, extract meaningful phonetic representations, and apply computational techniques for improved word retrieval. This section outlines the key steps in the correction process, from phonetic preprocessing to dimensionality reduction and similarity-based retrieval.

2.1 Phonetic Processing

The phonetic spelling correction system begins with preprocessing phonetic inputs, ensuring a structured representation suitable for computational analysis. The following subsections describe each stage of this process, including phoneme tokenisation, feature vectorisation and the application of dimensionality reduction to optimise computational efficiency.

2.1.1 Lexicon Selection

A corpus of **phoneme-to-word mappings** is required to serve as the reference dataset. Given the lack of a standard phonetic dataset for Australian English, we selected an IPA-based lexicon and structured phonemes as a discrete sequence $\mathbf{P} = (p_1, p_2, ..., p_n)$ where each phoneme p_i represents an atomic speech sound.

2.1.2 Phoneme Tokenisation

A **word**, denoted as w, is defined as a sequence of characters representing a discrete linguistic unit. In this context, words consist solely of alphabetic letters without spaces or punctuation, and their length varies depending on the input.

Each word is tokenised into *n*-grams (bigrams and trigrams) to capture phonetic structures. This process involves breaking the word into overlapping sequences of n consecutive characters. For example, the word *"hello"* is decomposed as follows:

$$\text{“hello”} \rightarrow \{\text{“}he\text{”}, \text{“}el\text{”}, \text{“}ll\text{”}, \text{“}lo\text{”}\} \quad \text{(bigrams)}$$

$$\text{“hello”} \rightarrow \{\text{“}hel\text{”}, \text{“}ell\text{”}, \text{“}llo\text{”}\} \quad \text{(trigrams)}$$

N-grams capture local patterns in phoneme sequences, which are crucial for distinguishing between different phonetic structures. *Bigrams* model direct adjacent phoneme relationships, while *trigrams* provide a

broader phonetic context. This multilevel representation enhances the model's ability to generalise across phonetic variations, improving similarity matching for phonetic spelling correction.

2.1.3 Phoneme-to-Letter Mapping

English orthography is non-phonemic, meaning that the spelling of words does not always correspond directly to their pronunciation. To address this, a mapping function $f : \mathbf{P} \to \mathbf{L}$ was developed, where f assigns phoneme sequences to plausible letter sequences. Some phonemes lacked direct English equivalents, necessitating the use of approximation mappings:

$$f(p_i) = \begin{cases} l_j, & \text{if } p_i \in \text{dom}(f) \\ \text{“UNK”}, & \text{otherwise} \end{cases}$$

In this function, l_j represents the letter sequence corresponding to the phoneme p_i, and "UNK" is used for phonemes without direct English equivalents.

Common phoneme sequences, such as those in words like *TRUNK* (/tɹʌŋk/), are explicitly accounted for in the mapping. The "UNK" token is only assigned when a phoneme has no plausible English representation or when a reasonable approximation does not exist. This ensures that frequently occurring phonemes are mapped accurately, while only rare or non-standard phonemes fall back to the unknown category.

2.2 Vectorisation of Phonetic Representations

Once the phonemes are mapped, each phonetic variant is transformed into a numerical vector for further processing.

2.2.1 *N*-Gram Feature Matrix

We construct an *n*-gram frequency matrix where **rows** represent words and **columns** correspond to *n*-gram features. The matrix in Table 1 captures the frequency of each *n*-gram within the word "hello".

Word	he	el	ll	lo	hl	ell
hello	1	1	1	1	0	1
halo	1	0	0	1	0	0

Table 1: *N*-Gram Frequency Matrix for Sample Word "hello"

For a given word w, its feature vector $\mathbf{x}_w$ is

$$\mathbf{x}_w = (x_1, x_2, \ldots, x_m) \in \mathbb{R}^m$$

where x_i represents the frequency of the i-th *n*-gram, and m represents the total number of possible bigrams and trigrams.

2.2.2 High-Dimensional Representation

The full dataset results in a high-dimensional space, where words are represented as points in $\mathbb{R}^m$. The *n*-gram feature vectors are constructed using non-negative integer counts, indicating the frequency of *n*-gram occurrences within each word. Although represented in $\mathbb{R}^m$, the feature vectors remain sparse, which means that most entries contain zeros.

This combination of high dimensionality and sparsity introduces additional challenges, as sparse data can lead to inefficiencies in both memory usage and similarity computations. To mitigate these issues, we apply dimensionality reduction techniques to the *n*-gram feature vectors, transforming them into a more compact representation that preserves the key features while improving computational efficiency.

2.3 Dimensionality Reduction

To mitigate high dimensionality, we applied Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). By reducing the dimensionality of the *n*-gram feature vectors, we make them more manageable and highlight the most important features.

2.3.1 Principal Component Analysis (PCA)

PCA transforms the data into a new coordinate system where the greatest variances are found along the first coordinates, known as the principal components. Given an input data matrix $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{n \times m}$ represents a dataset with n samples (rows) and m features (columns), PCA is performed on the covariance matrix:

$$\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

The eigenvectors of $\mathbf{C}$ define the principal components, and the corresponding eigenvalues indicate the amount of variance captured by each component. The transformation to the new coordinate system is given by:

$$\mathbf{Z} = \mathbf{X}\mathbf{W}$$

where $\mathbf{W} \in \mathbb{R}^{m \times k}$ is the matrix whose columns are the top-k eigenvectors of $\mathbf{C}$, corresponding to the largest eigenvalues. The resulting matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$ represents the transformed data in the reduced k-dimensional space.

2.3.2 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) factorises a given matrix into three other matrices, capturing the most significant underlying structures. Unlike PCA, which relies on the covariance matrix, SVD operates directly on the original data and is particularly useful for sparse matrices and text data. Given the n-gram matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, where n represents the number of samples (words) and m represents the number of features (*n*-grams), we decompose $\mathbf{X}$ as:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{U} \in \mathbb{R}^{n \times n}$ contains the left singular vectors, representing word relationships, $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ is a diagonal matrix of singular values, indicating the importance of each dimension, and $\mathbf{V}^T \in \mathbb{R}^{m \times m}$ contains the right singular vectors, representing *n*-gram relationships.

To reduce dimensionality while preserving essential structure, we approximate $\mathbf{X}$ using only the top-k singular values:

$$\mathbf{X}_k \approx \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$$

where $\mathbf{U}_k \in \mathbb{R}^{n \times k}$ contains the top-k left singular vectors, $\mathbf{\Sigma}_k \in \mathbb{R}^{k \times k}$ is the truncated diagonal matrix of the largest k singular values, and $\mathbf{V}_k^T \in \mathbb{R}^{k \times m}$ contains the top-k right singular vectors.

This reduced representation retains the most important phonetic structures while significantly reducing computational complexity.

2.4 Similarity Measures for Word Prediction

After dimensionality reduction, a new phonetic input is mapped into the reduced feature space, where its phonetic variant is compared to stored word representations. The goal is to retrieve the closest valid word based on phonetic distance.

2.4.1 Distance Metrics

Given a test phoneme input $\mathbf{q}$, similarity is computed against stored word vectors $\mathbf{X}_k$, which are precomputed from phonetic spellings. The similarity measures determine the most likely intended word.

A **distance metric** $d(\mathbf{q}, \mathbf{x})$ is a function that quantifies how different two phoneme vectors are. Formally, it must satisfy the following properties:

1. *Non-negativity*: $d(\mathbf{q}, \mathbf{x}) \geq 0$, with equality if and only if $\mathbf{q} = \mathbf{x}$.
2. *Symmetry*: $d(\mathbf{q}, \mathbf{x}) = d(\mathbf{x}, \mathbf{q})$.
3. *Triangular inequality*: $d(\mathbf{q}, \mathbf{x}) + d(\mathbf{x}, \mathbf{y}) \geq d(\mathbf{q}, \mathbf{y})$, ensuring that direct distances are never greater than indirect paths.

Since similarity is the inverse of distance, we define similarity measures as transformations of distance functions, such as:

$$s(\mathbf{q}, \mathbf{x}) = \frac{1}{1 + d(\mathbf{q}, \mathbf{x})}$$

or

$$s(\mathbf{q}, \mathbf{x}) = 1 - \frac{d(\mathbf{q}, \mathbf{x})}{\max d}$$

where higher similarity values indicate phonetic closeness. Cosine similarity is an exception, as it is inherently a similarity measure rather than a distance.

In this study, we evaluate phonetic retrieval using Cosine Similarity, Euclidean Distance, and Manhattan Distance, comparing their effectiveness in matching phonetic spellings to intended words.

1. **Cosine Similarity:** Measures the directional alignment of vectors, useful when phonetic representations vary in magnitude but maintain structural consistency:

$$\cos(\theta) = \frac{\mathbf{q} \cdot \mathbf{x}_w}{\|\mathbf{q}\|\|\mathbf{x}_w\|} = \frac{\sum_{i=1}^{n} q_i x_{w_i}}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} x_{w_i}^2}}$$

   where $\mathbf{q}$ is the test phoneme input vector and $\mathbf{x}_w$ is the stored word vector. This is effective for phonetic mappings where the relative relationships between features matter more than absolute distance.

2. **Euclidean Distance:** Measures the straight-line distance between phonetic vectors in the reduced space:

$$d_E(\mathbf{q}, \mathbf{x}_w) = \|\mathbf{q} - \mathbf{x}_w\|_2$$

   Useful for cases where phonetic misspellings alter multiple phoneme components simultaneously.

3. **Manhattan Distance:** Computes phonetic similarity by summing absolute differences:

$$d_M(\mathbf{q}, \mathbf{x}_w) = \sum_{i=1}^{k} |q_i - x_{w_i}|$$

   Captures phonetic structure by emphasising stepwise differences, making it ideal for cases where phoneme insertions, deletions, or substitutions occur in a structured, sequential manner - similar to navigating a grid where changes occur in discrete steps along individual phoneme dimensions.

2.5 Phonetic Spelling Correction and Word Retrieval

To find the best match, we retrieve the closest stored word w^*:

$$w^* = \arg\min_{w} d(\mathbf{q}, \mathbf{x}_w)$$

where d is the distance metric chosen. The process is as follows.

1. **Phonetic Vectorisation**: Encode the input word as an *n*-gram feature vector.
2. **Dimensionality Reduction**: Project the vector into the SVD/PCA-reduced feature space.
3. **Similarity Computation**: Find the closest stored phoneme representation based on similarity.
4. **Word Retrieval**: Retrieve the best match using SQLite, where phonetic variants are indexed for efficient lookup.

SQLite stores precomputed phoneme variants and their word mappings, allowing fast similarity-based retrieval rather than recomputing vector comparisons for every query.

By combining vector-based phoneme mapping, dimensionality reduction, and efficient similarity search, this approach effectively bridges the gap where traditional rule-based phonetic spelling correction methods fail.

2.6 Challenges in the Methodology

This section outlines the key challenges faced during the development of the phonetic spelling correction system and the solutions implemented.

2.6.1 Choosing the right Lexical Dataset

A phonetic lexicon was needed that included phoneme transcriptions to proceed. Several issues arose in finding the right lexicon. Many phonetic lexicons were based on American English, whereas the project focused on Australian English. In addition, different phonetic transcription systems, for example, ARPAbet vs. IPA, where ARPAbet lacks phonemes found in Australian English. An **Australian English lexicon** was selected and preprocessing techniques were applied to normalise the phoneme representation, ensuring spaces between phonemes for correct tokenisation.

2.6.2 Preprocessing Requirements: Formatting and Handling Special Characters

The lexicon contained various formatting inconsistencies. For instance, the phonemes had no spaces between them, making tokenisation difficult. Diacritics (“ ’ ”, “ , ”, and “ : ”) were removed. The entries were cleaned of zero-width joiners and other unicode characters. The solution in implementing these preprocessing steps was to strip stress markers and ensure spaces between phonemes for better tokenisation. The unwanted Unicode characters were removed.

2.6.3 Data Explosion Issue: Exponential growth in data set size

The generation of phoneme variants led to an exponential increase in the size of the data set. Each phoneme sequence had multiple possible letter mappings, resulting in a combinatorial expansion of word spellings. As the number of phonemes per sequence increased, the number of possible spellings grew exponentially, significantly increasing the storage and computational requirements. For instance, if a phoneme sequence contained five (5) phonemes and each phoneme had three (3) possible letter mappings, the number of unique spellings for a single word would be:

$$3^5 = 243$$

This rapid combinatorial growth underscores the need for dimensionality reduction and efficient phonetic encoding techniques to manage the dataset effectively. The solution involved restricting phoneme-letter mappings to a maximum of 10 variants per word, implementing sampling techniques to limit phoneme sequence variations, and queue processing when saving in SQLite to avoid memory overload.

2.6.4 Mapping Issues: Converting Phonemes to Letters

There were many non-singular one-to-one phoneme mappings, meaning that multiple letters could map to the same phoneme. For example, “f”, “ph”, “gh” all map to IPA [f] and some phonemes had no direct letter

equivalent in English. Unknown phonemes appeared that were not in the mapping dictionary. For example, the voiceless velar fricative /x/ as in Scottish English "loch" or German "Bach", has no standard English letter, and for the glottal stop /ʔ/ in Cockney "bo'le" for "bottle", English does not have a letter at all. The solutions involved creating a phoneme-to-letter dictionary with multiple mappings per phoneme. Then, to identify and capture phonemes that had not been converted, introducing a default "UNK" (unknown) mapping to log unhandled phonemes to a separate file (`unhandled_phonemes.log`) for further debugging and mapping. For instance, mapping /x/ → ["kh", "h", "k"] and mapped the glottal stop /ʔ/ → ["", ""]. Additionally, tokenisation methods were implemented to preserve multicharacter phonemes before mapping aided correct phoneme-to-letter transcription.

2.6.5 Vectorisation and Dimensionality Challenges: High-Dimensional Data for Phoneme Embeddings

After mapping phonemes to letters, the next challenge was to vectorise phoneme spellings. The system used bigram and trigram vectorisation, which created a high-dimensional feature space, increasing computational cost. To solve this, PCA (Principal Component Analysis) and SVD (Singular Value Decomposition) were applied to reduce dimensions. Variance retention in scaled versus unscaled data was evaluated to ensure optimised results. Dimensionality was successfully reduced while maintaining the distinctiveness of the word. The Manhattan distance, cosine similarity, and Euclidean distance were used to compare phonetic spellings.

2.6.6 Summary of Key Challenges and Solutions

The challenges in this project influenced the design of the phonetic spelling prediction system, and are summarised in Table 2. Addressing these issues required:

- **Selecting the right dataset** (IPA-spaced phonetic lexicon).
- **Implementing strong preprocessing** (cleaning, tokenisation).
- **Optimising computational efficiency** (restricting explosion, reducing dimensions).
- **Testing multiple approaches** (phoneme-letter mapping, vectorisation).

These solutions ensured that the project successfully processed phonetic spellings while maintaining computational feasibility.

3 Results

This section presents the impact of pre-scaled and post-scaled dimensionality reduction on phonetic spelling correction. The focus is on variance retention and the effectiveness of similarity measures.

Challenge	Issue	Solution
Finding a lexicon	No standard dataset for **Australian English phonetics**	Selected **Australian lexicon** and applied **IPA normalisation**
Preprocessing phonemes	**No spaces, stress markers, Unicode issues**	Used **regex-based cleaning**, ensured **space-separated phonemes**
Data Explosion	Phoneme-to-letter mappings **increased dataset size massively**	**Limited mappings**, capped **variant generation**, used **chunked processing**
Phoneme-to-letter mapping	Some phonemes **had no English letter equivalent**	Created **fallback "UNK" mappings, logged missing phonemes**
Vectorisation complexity	**Bigram/trigram embeddings** created **high-dimensional vectors**	Applied **PCA/SVD**, optimised **variance retention**
Dimensionality reduction	Needed to balance **information loss vs. vector size**	Used **variance analysis** before and after scaling

Table 2: Summary of Challenges and Solutions in Phonetic Spelling Correction

3.1 Dimensionality Reduction and Variance Analysis

Dimensionality reduction was applied to retain the essential phonetic structure while reducing computational complexity. PCA and SVD were applied before and after feature scaling, and their effectiveness was analysed based on variance retention.

3.1.1 Variance Per Component

The variance explained per component for PCA and SVD is shown in Figure 1.

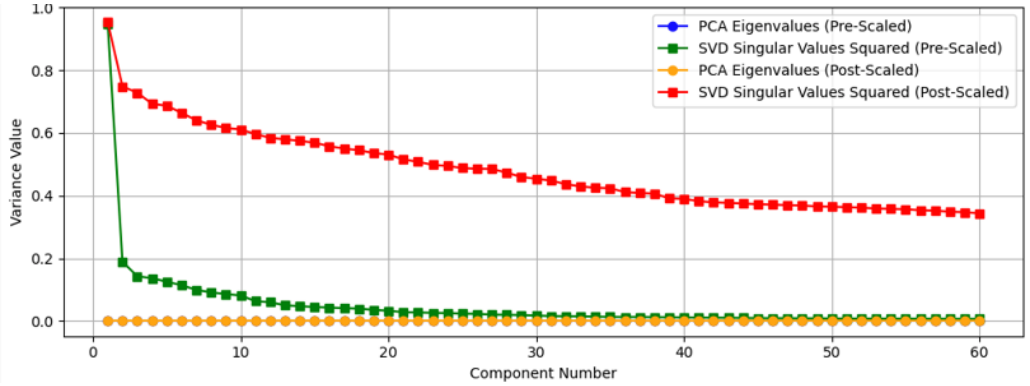


Figure 1: Per-component variance for PCA and SVD (pre- and post-scaling).

From the figure, it is evident that pre-scaled SVD captures most variance in the first few components, making it efficient for compression. In contrast, post-scaled SVD distributes variance more evenly, avoiding dominance by high-magnitude phonemes.

3.1.2 Cumulative Variance Explained

The cumulative variance for PCA and SVD is visualised in Figure 2.

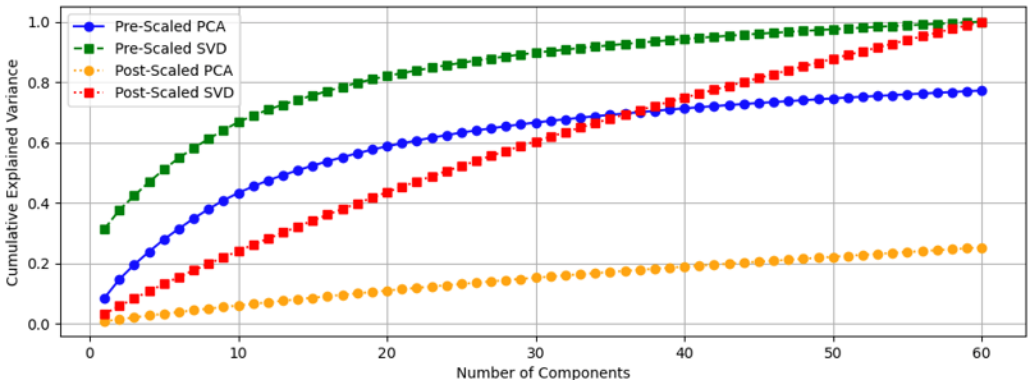


Figure 2: Cumulative variance explained for PCA and SVD (pre- and post-scaling).

Observations from the figure indicate that pre-scaled SVD reaches 80% variance at approximately 20 components, while post-scaled SVD requires around 45 components. Despite the slower variance accumulation, post-scaled SVD leads to better word retrieval accuracy, as discussed next.

3.2 Effectiveness of Similarity Measures in Phonetic Word Prediction

Phonetic similarity retrieval was evaluated using cosine similarity, Euclidean distance, and Manhattan distance. The results were compared using both pre-scaled and post-scaled datasets.

3.2.1 Pre-Scaled SVD Results

Cosine similarity retrieved phoneme variants with some structure but struggled with multi-syllabic words. Euclidean and Manhattan distances disproportionately matched shorter phoneme variants, making them ineffective for accurate phonetic spelling correction. Many matches were single-phoneme outputs, leading to errors in phonetic word prediction.

Table 3: Phonetic Word Prediction (Pre-Scaled SVD)

Input	Method	Closest Match	Distance
ingre	Cosine	surly, swirly, surlier, surreal	≈ 0.55
	Euclidean	e, ea, E	≈ 1.32
	Manhattan	ER, Er, Ir	≈ 7.27
halow	Cosine	revers's, Rickey's, air	≈ 0.56
	Euclidean	A, AA, AR	≈ 0.44
	Manhattan	A, AA, AR	≈ 2.42
pitcha	Cosine	Quebec's, weep, quickie, keep	≈ 0.63
	Euclidean	E, e, ea	≈ 1.18
	Manhattan	A, a, eh	≈ 6.80

3.2.2 Post-Scaled SVD Results

A significant improvement was observed in post-scaled SVD, particularly for cosine similarity. Cosine similarity now retrieves multi-syllabic phoneme variants, indicating better phonetic structure preservation. Euclidean and Manhattan distances still fail to capture the phonetic structure, returning mostly short, truncated phoneme variants. The retrieved words are now semantically and phonologically closer to the intended correct spelling.

Table 4: Phonetic Word Prediction (Post-Scaled SVD)

Input	Method	Closest Match	Distance
ingre	Cosine	tinkering, ringings, printings	≈ 0.09
	Euclidean	e'en, T, Thai	≈ 4.17
	Manhattan	T, Te, Ty	≈ 22.97
halow	Cosine	Missourian, Rousseau's, pronoun	≈ 0.05
	Euclidean	Ron, rain, rein	≈ 3.83
	Manhattan	T, Te, Thai	≈ 20.36
pitcha	Cosine	chitchat, Rorschach, nature	≈ 0.01
	Euclidean	Ron, Wren, rain	≈ 3.95
	Manhattan	T, Te, Thai	≈ 21.63

3.3 Mathematical Interpretation of Results

The results suggest that post-scaled SVD improves the phoneme vector space for word retrieval due to balanced feature scaling. Cosine similarity measures the angular distance, meaning that it performs best when feature magnitudes are normalised. Euclidean and Manhattan distances depend on absolute magnitude, making them unsuitable when phoneme representations vary significantly in scale.

Mathematically, post-scaling transforms feature distributions to have unit variance, ensuring that phonetic distance is based on meaningful phoneme structure rather than raw frequency. Cosine similarity benefits from this transformation because it measures the relative orientation of phoneme vectors rather than their raw magnitude.

4 Discussion and Conclusion

4.1 Summary of Findings

This study explored the correction of phonetic spelling using dimensionality reduction techniques and similarity metrics. The primary goal was to improve the retrieval of phonetic words using feature transformation methods such as PCA and SVD. The performance of pre-scaled and post-scaled SVD was evaluated based on variance retention and phonetic similarity measures.

Key results demonstrated that:

- Pre-scaled SVD retained a higher cumulative variance in fewer dimensions but led to suboptimal word retrieval due to imbalanced feature scaling.
- Post-scaled SVD, despite requiring more components to reach the same variance threshold, significantly improved phonetic word prediction accuracy.
- Cosine similarity emerged as the most effective metric, outperforming Euclidean and Manhattan distances in retrieving multi-syllabic phoneme variants.

The shift from pre-scaled to post-scaled SVD improved the alignment of phoneme vector representations, leading to more accurate phonetic word retrieval. Although the Euclidean and Manhattan distances struggled with phonetic structure preservation, cosine similarity successfully captured phonological relationships by measuring angular similarity.

4.2 Implications and Contributions

The results highlight the importance of feature scaling in phonetic representation. The post-scaling process ensures that phoneme vectors are mapped in a way that preserves their relative phonetic structure, improving similarity-based retrieval. The findings contribute to enhanced phonetic spelling correction methodologies, improved handling of non-standard phoneme spellings, and a scalable approach to processing phonetic input using dimensionality reduction.

This research provides a foundation for further refinement of phonetic spelling correction models, particularly in applications where spelling errors are heavily influenced by phonetic transcription inconsistencies.

4.3 Future Work

Although this study demonstrates the effectiveness of post-scaled SVD and cosine similarity, several enhancements can be pursued. Integrating Hidden Markov Models (HMMs) could improve phoneme clustering by capturing sequential dependencies in spoken language. Further, exploring contextual word embeddings, for instance Word2Vec, BERT or transformer-based architectures, could refine phonetic similarity by incorporating broader linguistic patterns.

Expanding the dataset with larger and more diverse phonetic corpora would enhance model robustness and generalisation, ensuring better performance across various dialects and speech patterns. Furthermore, alternative dimensionality reduction techniques, such as t-SNE and UMAP, could be explored to improve the preservation of phonetic structures in a reduced space.

Deploying this methodology in a real-time phonetic spelling correction system and conducting user evaluations would provide insights into its practical effectiveness. Future iterations could incorporate feedback-driven refinements to optimise accuracy and usability in real-world applications.

4.4 Conclusion

This study demonstrated that post-scaled SVD, combined with cosine similarity, significantly enhances phonetic spelling correction. While pre-scaled SVD efficiently captures variance, its reliance on unnormalised feature magnitudes reduces retrieval accuracy. In contrast, post-scaling ensures balanced phoneme representations, leading to more precise word predictions.

These findings highlight the crucial role of feature scaling in phoneme vectorisation and confirm that cosine similarity is the most effective metric for phonetic retrieval. By preserving phonetic structure while reducing dimensionality, post-scaled SVD provides a strong foundation for real-world phonetic spell-checking systems.

Future research should refine phoneme embeddings, incorporating probabilistic models, contextual embeddings, or deep learning architectures to further enhance accuracy. Integrating this approach into real-time phonetic correction tools could provide practical support for individuals with phonetic spelling tendencies, improving accessibility and communication.

Acknowledgements

- Associate Professor Simon James
- Associate Professor Julien Ugon

With thanks

References

Bahr, Ruth Huntley, Stephanie Lebby, and Louise C Wilkinson (2020). "Spelling error analysis of written summaries in an academic register by students with specific learning disabilities: Phonological, orthographic, and morphological influences". In: *Reading and Writing* 33.1, pp. 121–142.

Brown, Gordon DA and Richard PW Loosemore (1994). "Computational approaches to normal and impaired spelling". In: *Handbook of spelling: Theory, process and intervention*, pp. 319–335.

Dessemontet, Rachel Sermier et al. (2021). "Effects of a phonics-based intervention on the reading skills of students with intellectual disability". In: *Research in Developmental Disabilities* 111, p. 103883.

Flint, Emma et al. (2017). "A text normalisation system for non-standard English words". In: *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 107–115.

Frith, Uta (1980). *Cognitive processes in spelling*. ERIC.

Garay-Vitoria, Nestor and Julio Abascal (2006). "Text prediction systems: a survey". In: *Universal Access in the Information Society* 4, pp. 188–203.

Knuth, Donald E. (1998). *The Art of Computer Programming, Volume 3: Sorting and Searching*. 2nd. Addison-Wesley. ISBN: 978-0201896855.

Leutzinger, Bridget (2022). "Phonics and Spelling Intervention for Children with Intellectual Disabilities". In: *Public Access Theses, Dissertations, and Student Research from the College of Education and Human Sciences. 412*.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *arXiv preprint arXiv:1301.3781*.

Nagata, Ryo, Hiroya Takamura, and Graham Neubig (2017). "Adaptive spelling error correction models for learner English". In: *Procedia Computer Science* 112, pp. 474–483.

Pan, Steven C, Timothy C Rickard, and Robert A Bjork (2021). "Does spelling still matter—and if so, how should it be taught? Perspectives from contemporary and historical research". In: *Educational Psychology Review*, pp. 1–30.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Philips, Lawrence (2000). "The double metaphone search algorithm". In: *C/C++ users journal* 18.6, pp. 38–43.

Phillips, Lawrence (2000). *The Double Metaphone Search Algorithm*. URL: `https://drdobbs.com/the-double-metaphone-search-algorithm/184401251?pgno=2`.

Rubehn, Arne et al. (2024). "Generating Feature Vectors from Phonetic Transcriptions in Cross-Linguistic Data Formats". In: *arXiv preprint arXiv:2405.04271*.

Ryskina, Mariia (2022). "Learning Computational Models of Non-Standard Language". PhD thesis. Carnegie Mellon University.

Sofroniev, Pavel and Çağrı Çöltekin (2018). “Phonetic vector representations for sound sequence alignment”. In: *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 111–116.

Tong, Xiuli et al. (2009). “Morphological awareness, orthographic knowledge, and spelling errors: Keys to understanding early Chinese literacy acquisition”. In: *Scientific Studies of Reading* 13.5, pp. 426–452.

Treistman, Avraham et al. (2022). “Word embedding dimensionality reduction using dynamic variance thresholding (DyVaT)”. In: *Expert Systems with Applications* 208, p. 118157.

Trinh, Ha et al. (2012). “Applying prediction techniques to phoneme-based AAC systems”. In: *Proceedings of the Third Workshop on Speech and Language Processing for Assistive Technologies*, pp. 19–27.

Vykhovanets, Valeriy S, Jianming Du, and Sergey A Sakulin (2020). “An overview of phonetic encoding algorithms”. In: *Automation and Remote Control* 81, pp. 1896–1910.

Zouhar, Vilém et al. (2023). “Pwesuite: Phonetic word embeddings and tasks they facilitate”. In: *arXiv preprint arXiv:2304.02541*.