

# Phonetic Representation for Language Modeling with Rhyme

**Jaiv Doshi**  
jpdoshi@usc.edu

**Nathan Sparks**  
nsparks@usc.edu

**Daniel Youn**  
younh@usc.edu

## Abstract

In this paper, we first examine the relative lack of phonetic representation being used for language modeling compared to orthographic representation. We then propose novel metrics for evaluating rhymes across multiple rhyme types, and compare the effectiveness of these metrics to rhyme evaluation metrics developed by others. These metrics employ conversion to a phonetic representation to obtain an evaluation grounded in word sound. Next, we test the effectiveness of a state of the art LLM on a rhyming task, and compare its performance when various prompting techniques are used. Finally, we propose a method for analyzing the influence of phonetic and orthographic representation on language model performance, specifically on a task with a strong phonetic component, rhyming.

## 1 Introduction

A great deal of language model development has been based in orthographic language representation, both in the datasets used for training and in the tasks focused on by the development. Meanwhile, LM support for phonetic representation such as the International Phonetic Alphabet (IPA) is very poor (Gale et. al., 2023). The focus on orthographic over phonetic representations is driven by multiple factors.

The foremost factor is data availability. Huge amounts of English text data are widely available from both the internet and literature, whereas data with an orthographic representation is quite limited by comparison. The nature of the tasks being researched is also relevant. Tasks like spell check and summarization have little to no need for phonetic representation and thus can be explored without exiting the orthographic domain. The encoding, too, plays an important role in language model development. Most common encoding schemes like byte-pair encoding (BPE) encode words at the

word, subword, or character level and are thus inherently orthographic.

Ultimately, this focus on orthographic representation is not without its drawbacks. It is known that “neural language models are able to capture implicit information about phonology from orthography”. However, this implicit information is limited by encoding, training, and embeddings (Elman, 1990; Prince and Smolensky, 1997). Thus, ignoring the phonetic component of language may be harming the ability of LMs to perform well on tasks with a phonetic component such as rhyming and analyzing pronunciations.

Furthermore, despite the remarkable advancements in general language generation by recent large language models (LLMs), there still needs to be a noticeable improvement in their ability to produce rhyming patterns reliably (Popescu-Belis et al., 2023).

## 2 Background

Language inherently consists of both orthographic, or text-based components and phonetic, or sound-based components. These components are closely connected with the written word having corresponding pronunciations, and the spoken word having corresponding transcriptions. Despite this connectivity, orthographic and phonetic components cannot be consistently inferred from each other. This is demonstrated by the nontriviality of pronouncing an unseen written word or transcribing an unseen spoken word. English in particular is often referred to as being “not phonetic”, meaning that the inference from orthographic to phonetic representation within the English language is difficult and inconsistent.

### 2.1 Graphemes and Phonemes

Orthographic and phonetic representations each have their own unit of language representation, known as graphemes and phonemes, respectively.

- Grapheme: the smallest meaningful contrastive unit in a writing system
- Phoneme: the smallest unit of speech distinguishing one word (or word element) from another

A grapheme is typically a single character. Examples of phonemes can be seen in the appendix. Grapheme representation and phonemic representation are used to represent the orthographic and phonetic components of language, respectively.

The phonemes we use here are elements of a set of phonetic transcription codes called ARPAbet. ARPAbet describes phonemes that are a part of General American English with unique sequences of uppercase characters in the English alphabet. The phonemes of vowels can optionally be followed by a numeric stressor with values ranging from 0 to 2 inclusive that describe how the vowel is accented.

### 3 Related Work

BORT (Gale et al., 2023) explores phonemic language modeling using an LLM that accepts a mixture of English pronunciations in IPA and English orthography and demonstrates its use on a task motivated by a real-world clinical problem: analysis of speech errors made by individuals with aphasia. The researchers create the BORT models by extending the pre-training of an existing LLM, BART. Their self-supervised task focuses on a novel IPA-to-orthography translation task given a document, they transform some words into IPA, then train the model to restore the orthography. The pretrained BORT model is available for anyone to download and fine-tune.

Meanwhile, previous rhyme generation work explores training on a corpus of a poetry form, rap lyrics, or generic prose. Notably, these approaches use annotation techniques, such as specifying rhyming words or line endings, instead of using the phonemic representation of words. The Mechanical Bard (Agnew et. al., 2023) consider the automated generation of sonnets using a constrained decoding approach within preset poetic constraints (e.g., number of syllables, rhyme scheme, and pentameter). Additionally, their method includes 120 handcrafted grammar templates that encode the part-of-speech structure of a line of poetry. They fine-tune GPT-2 on a large corpus of over 15000 poems and a smaller corpus of sonnets, and apply

their decoding generation procedure on generated text. Human evaluation is used to confirm that their approach produces sonnets that resemble human authored sonnets. The code for The Mechanical Bard is available on GitHub. GPoeT (Popescu-Belis et al., 2023) is trained on synthetic quatrains with consecutive rhymes (AABB) and alternating rhymes (ABAB). This approach obtains valid rhyming in nearly 60% of consecutive lines and 45% of alternating lines in generated samples. PoeLM (Ormazabal et al., 2022) learns rhythm and syllables from a large corpus of Spanish and Basque prose. This model learns control tags, which specify the length, end rhyme, and last syllable of every sentence, and leverages them to generate lines of poetry. This approach is evaluated by asking human judges if they prefer PoeLM generated or human generated poems.

## 4 Dataset

For the modeling experiment task we needed our dataset to consist of rhyming line pairs. This kind of data was difficult to find because while there are many open source datasets that contain rhyming data, it would be difficult to extract the lines that rhyme with each other. In the end, we decided to use a dataset called the “Chicago Rhyming Poetry Corpus” that consists of annotated poetry across various authors. This dataset consists of 4MB of English text with 130K total lines and has each stanza annotated with the rhyming scheme (Lenzo, 2023).

### 4.1 Parsing Rhyming Lines

The rhyme schemes provided a convenient way to match lines with another line that rhymes with it. Each stanza in the dataset begins with a line that defines the rhyming scheme. The line starts with the word “RHYME” indicating that this line defines a rhyming scheme. Then follow a series of letters. The first letter represents the first line, the second letter represents the second line, and so on (Lenzo, 2023). If two letters are the same, it means that the lines that map to those letters rhyme. For example: “RHYME a b a b” means that the first and third lines rhyme and the second and fourth lines rhyme. If the stanza is too long, the rhyming scheme ends with “\*” indicating that the rhyming scheme repeats. For example, “RHYME a a \*” means that the first two lines, second two lines, and each pair of lines that remain in the stanza rhyme.

## 4.2 Grapheme to Phoneme

Now that we have the grapheme representation of our data, we need to translate them into their phonemic representations. There are two main ways of doing this. One way is to use The Carnegie Mellon University (CMU) Pronouncing Dictionary. This dictionary consists of the pronunciations of over 134,000 English words represented with a space separated string of phonemes. For example, the word “language” would be mapped to “L AE1 NG G W AH0 JH” (Popescu-Belis et al., 2023). The other way is to use a deep learning seq2seq framework from a python model called g2pE that predicts the best pronunciation of the word and returns a space separated string of phonemes just as The CMU Pronouncing Dictionary does (Kyubyong, 2019). This model was especially useful because our dataset contained many words from Old English that are no longer valid words in modern English. The downside of using the g2pE model is that it takes around half a second for it to return its prediction. In order to optimize the speed at which we translate our grapheme dataset to phonemes, we first create a map that maps all unique words in our dataset to their phonemic representations. After creating this dictionary, we are able to have constant time lookup to the phonemic representations of any word in our dataset.

With these tools available, we can now convert all line pairs in our grapheme dataset to phonemes. However, we cannot simply concatenate the phonemic representations of words together. Because the phonemic representations of words are a space separated string of phonemes, we will not be able to distinguish the start and ends of words if we just concatenate them together. For example, consider the sentence “Hello world.” If we were to simply concatenate the phonemic representations, we would have “HH AH L OW W ER L D,” but it is ambiguous where one word begins and the next starts. To fix this issue we treat each phoneme as a token and change the phonemic representation of a word from a space separated string of phonemes to a list of the phonemes in order. Then to produce the phonemic representation of a line, we join the phonemic representations of words together with spaces between them. For our example sentence “Hello world,” it would now become [“HH”, “AH”, “L”, “OW”, “W”, “ER”, “L”, “D”]. This format makes it clear for both humans and machines where the cutoff between words are.

## 5 Evaluating Rhymes

A criterion for measuring the number of rhyme pairs is key for the present study. We present a metric that distinguishes between perfect rhymes, assonant rhymes, consonant rhymes, and no rhymes, using the CMU Pronouncing Dictionary. We test it on a custom modified corpus of human poetry annotated for rhyme and show that its accuracy is sufficient for use in this study.

### 5.1 Preliminary Analysis

An initial approach is to assess the similarity of the last K phonemes of words in a rhyme pair. This method focuses on the terminal sounds of words. As such, it is particularly useful for capturing rhymes where the final sound is the same between both words.

To illustrate, consider the couplet from Richard Lovelace’s “La Bella Bona Roba”:

Ye cloudy spark lights, whose vast multi-  
tude,  
Waite on this star in her first magnitude.

After removing punctuation and transcribing the final words into phonemes using CMUDict, we obtain “M AH L T AH T UW D” for ‘multitude’ and “M AE G N AH T UW D” for ‘magnitude’. Comparing the last two phonemes (“UW D”) of each word reveals a 100% match, while comparing the last five phonemes shows an 80% match.

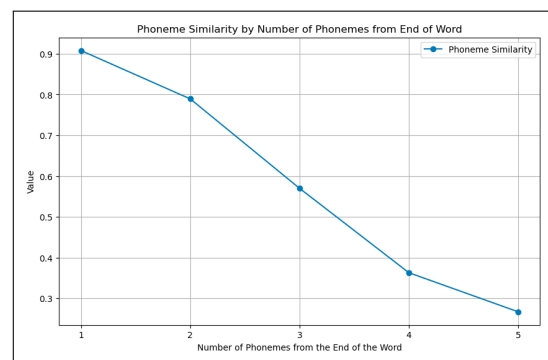


Figure 1: Phoneme Similarity by Number of Phonemes from End of Word

Applying this metric on our modified Chicago Poetry Corpus, yielded sufficient initial results. When evaluating the similarity of the last phoneme alone, the metric achieved a 90% accuracy rate. However, accuracy diminished with the inclusion

of more phonemes, a trend in line with our expectations.

Despite its efficacy, this metric exhibits limitations in identifying imperfect rhymes, such as “Eye” and “Light” (phonemes "AY" and "L AY T"). Although they share the "AY" sound, because this sound is not in the same position from the end in both words, our metric registers 0% similarity. To address this limitation, we introduce a more sophisticated algorithm in the following section, which is specifically designed to capture these nuanced aspects of poetic rhyme.

## 5.2 Definitions of Rhymes

GPoeT (Popescu-Belis et al., 2023) formalized an initial set of definitions for rhymes. They defined a perfect rhyme as the identity of the final vowel and consonant sound of a word, starting with the first vowel of the last stressed syllable and an assonant rhyme as the identity of the final vowels in the last stressed syllable, but not of the ending consonant.

To broaden the amount of available candidates for a rhyme, they excluded stress information from phonemes. Additionally, they distinguish 15 phonemic vowels (e.g., ‘AH’, ‘AW’, ‘EY’, ‘OY’) and consider all other phonemes as consonants.

An example of perfect rhymes are the words "Light" and "Night" (phonemes "L AY T" and "N AY T"). Both words share the same ending vowel sound “AY” followed by the same consonant sound “T”. In contrast, an example of assonant rhymes are the words "Release" and "Feast" (phonemes "R IY L IY S" and "F IY S T"). Both words share the same ending vowel sound “IY”, but they are followed by different consonant sounds “S” and “S T”.

These definitions, however, do not encompass the full spectrum of rhyme types. In our improved definition, we incorporate consonant rhymes, drawing from Peter Dale’s "An Introduction to Rhyme," which explores the evolution of rhymes over the century (Dale 1998).

An example of consonant rhymes are the words “Spot” and “Cut” (phonemes "S P AA T" and "C AH T"). Here, the words terminate with different vowel sounds "AA" and "AH", but they follow with the same consonant sound "T".

Building on this, we propose a revised definition for rhymes between two words,  $w_1$  and  $w_2$ , based on their phonetic representations,  $phon(w_i)$ . Our criteria are as follows:

- We have a perfect rhyme if  $phon(w_1)$  and  $phon(w_2)$  end with the same vowel followed by the same consonant(s), if any.
- We have an assonant rhyme if  $phon(w_1)$  and  $phon(w_2)$  end with the same vowel, followed by one or more non-identical consonants.
- We have a consonant rhyme if  $phon(w_1)$  and  $phon(w_2)$  end with different vowels, followed by the same consonants.
- Otherwise, the lines do not rhyme.

This refined definition allows for a more comprehensive and nuanced analysis of rhymes in poetry, catering to the diverse forms of rhyming present in contemporary literature.

## 5.3 Validating our Metric

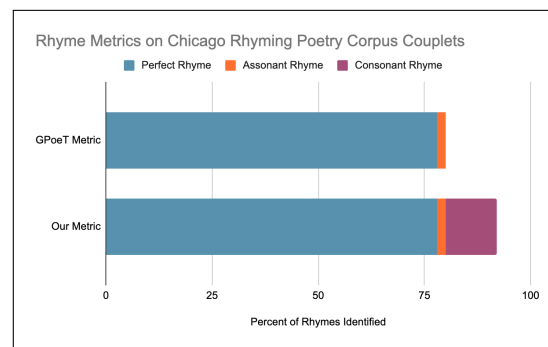


Figure 2: Rhyme Metrics on Chicago Rhyming Poetry Corpus

Our proposed metric for rhymes follows from the aforementioned definitions. We validated our metric on our modified Chicago Poetry Corpus and compared it against the metric used in the GPoeT paper.

Both metrics demonstrate an equivalent capacity in identifying perfect and assonant rhymes, as evidenced by the identical scores of 78% for perfect rhymes and 2% for assonant rhymes. However, a notable distinction emerges in the detection of consonant rhymes. Our metric successfully identified 12% of the rhymes as consonant rhymes, a category in which the GPoeT Metric did not register any. This difference underscores the enhanced sensitivity of our metric. It effectively identifies 5636 examples of consonant rhymes, which are overlooked by the GPoeT method. As such, we successfully demonstrate the robustness of our metric in encompassing a wider range of rhyme types.

## 5.4 Potential Issues

In our analysis, we encountered several challenges which may contribute to the less than perfect scores observed. One such challenge stems from the inherent complexities of the ARPABET system, particularly in its representation of phonemes. For instance, in the case of words like 'Heard' (HH ER1 D) and 'Yard' (Y AA1 R D), the ARPABET merges the 'R' sound into the adjacent vowel sounds, such as 'ER'. This merging makes it difficult to separate the vowel and consonant sounds distinctly, challenging our metric's ability to recognize the similarity in the 'RD' consonant sound at the end of these words.

Another issue arises from inaccuracies in grapheme-to-phoneme (G2P) conversions, especially for words not included in the CMU Pronunciation Dictionary. A notable example is the word 'Behoof', which our G2P model converts incorrectly (B EH1 HH UH2 D), erroneously suggesting it ends with a 'D' consonant sound. This incorrect conversion leads to inaccurate rhyme comparisons, as seen when paired with the word 'Proof' (P R UW1 F).

Furthermore, the strict definitions of rhymes pose their own challenges. Consider the word pair 'Commitment' (K AH M IH T M AH N T) and 'Identical' (AY D EH N T IH K AH L). According to our definition of assonant rhymes — where both words end with the same vowel sound followed by different consonants — this pair fits the criterion, sharing the 'AH' vowel sound. However, intuitively, 'Commitment' and 'Identical' do not seem to rhyme. This discrepancy highlights that while some consonant pairs may sound alike, others do not. For example, the consonant pairs 'N T' and 'L' in this instance do not produce a similar sound. Addressing this in future work, we might consider integrating principles from the Mnemonic Major System to better define consonant similarity.

These examples underscore the complexity in rhyme detection and validation. The evolution of pronunciation and the imperfections inherent in the annotation process further complicate the reliability of human assessments of rhymes. This variability in linguistic interpretation underscores the need for continual refinement in our metric.

## 6 Prompting

In our investigation, we evaluated the proficiency of GPT-3.5 Turbo in generating subsequent rhyming

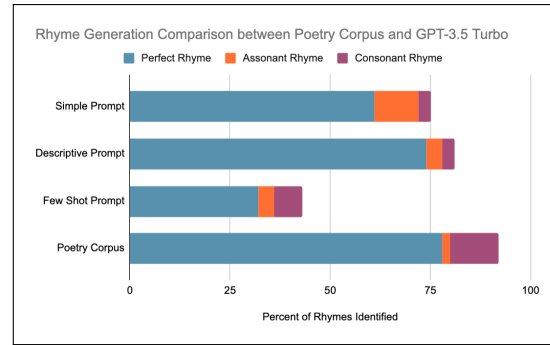


Figure 3: Rhyme Generation Comparison between Poetry Corpus and GPT-3.5 Turbo

lines in poetry. We conducted experiments with the first 100 examples in our modified corpus. For each test, GPT-3.5 Turbo was given a prompt to generate the next line, followed by the first line of a couplet.

The prompts varied in complexity:

- Simple Prompt: "Continue the poem with the next rhyming line."
- Descriptive Prompt: "Continue the poem with the next rhyming line. Ensure that the last word of the generated line rhymes with the last word of the provided line."
- Few Shot Prompt: Similar to the descriptive prompt, but supplemented with an example couplet for each type of rhyme. Full prompt provided in the appendix.

Detailed examples of outputs from all three styles of prompts are provided in the appendix.

The simpler prompts achieved higher success with perfect rhymes, suggesting GPT-3.5 Turbo's inherent capability to follow basic rhyming structures. However, when it came to more nuanced rhyme types like assonant and consonant rhymes, the model struggled, potentially due to the complexity and variability inherent in these rhyme types.

Notably, the Few Shot Prompt method underperformed across all rhyme types. Our analysis indicates that this might be due to the model's interpretation of the initial line as an incomplete fragment, leading it to first attempt to complete the provided line with its own rhyme before generating a subsequent line. This often resulted in a mismatched rhyme pair. Furthermore, it occasionally generated lines that did not rhyme at all or tried to semantically complete the initial sentence rather than focusing on the rhyming structure.

One hypothesis for these discrepancies is that GPT-3.5 Turbo, given only the first line, aims to create a contextually and semantically coherent next line. This differs from the original poetry, which might have been extracted from a larger piece, providing more context for the rhyme. The AI model’s focus on contextual coherence could interfere with its ability to strictly adhere to the rhyme pattern, particularly in more complex rhyming structures.

These findings suggest that while GPT-3.5 Turbo shows promise in generating rhyming lines, particularly for simpler rhyme schemes, its performance varies significantly based on the prompt structure and the complexity of the rhyme type.

## 7 Modeling Experiment

The goal of the modeling experiment is to examine how grapheme and phoneme representation affect the ability of a language model to learn a phonetic task. We hypothesize that phonetic representation will improve model performance on a task of generating a rhyming line based on an input line.

### 7.1 Experiment Setup

To test the hypothesis, we aimed to compare two analogous models: one model trained with a grapheme representation at the character level and another model trained with a phoneme representation using ARPAbet. The nature of the task is generation of an output sequence which rhymes (by end word) with an input sequence. Thus, we chose to use a sequence-to-sequence (seq2seq) model architecture with a LSTM encoder and decoder. See appendix for model diagram. Rhyming line pairs extracted from the Chicago Rhyming Poetry Corpus are used for training and testing of the model.

### 7.2 Results

Training the grapheme model on the rhyming line pairs resulted in near identical output sequences being generated for any input sequence we tested on. We tried training the model several times with varying numbers of epochs. Each time though, the model converged to generating the same output sequence for any input sequence. Similar results were seen from training the phoneme model. Thus, the experiment was unsuccessful since we could not compare the ability of the models to generate rhymes. The uniform generation indicates a failure of the model to learn the variation in the data. We believe there are three main reasons for these

results.

Firstly, each input sentence is only seen once, meaning that the significance of the rhyming end word is less apparent than if there were several examples in the training of possible rhyming output lines. We could try to fix this by augmenting the data with more output examples for each input line. Secondly, the task of generating a rhyming output line is both open-ended for the body of the line and close-ended for the end word which must rhyme. Such a task may be inherently difficult to learn, especially without a very high number of examples.

Thirdly, the size of our model was limited by compute resources, which may have prevented it from learning the variation in the data well. Increasing our model size could aid the process of learning the complex relationship between sequences.

## References

- Agnew, E., Qiu, M., Zhu, L., Wiseman, S., and Rudin, C. (2023). The Mechanical Bard: An Interpretable Machine Learning Approach to Shakespearean Sonnet Generation. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 1627–1638, Toronto, Canada. Association for Computational Linguistics.
- Dale, P. (1998). An Introduction to Rhyme. Bellew Publishing Co Ltd.
- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.
- Gale, R., Salem, A., Fergadiotis, G., and Bedrick, S. (2023). Mixed Orthographic/Phonemic Language Modeling: Beyond Orthographically Restricted Transformers (BORT).
- Kyubyong. (2019, December 31). Kyubyong/G2p: G2p: English grapheme to phoneme conversion. GitHub. <https://github.com/Kyubyong/g2p>
- Lenzo, K. (n.d.). The CMU Pronouncing Dictionary. Carnegie Mellon Speech Group. <http://www.speech.cs.cmu.edu/cgi-bin/cmudict?in=world>
- Popescu-Belis, A., Atrio, À. R., Bernath, B., Boisson, E., Ferrari, T., Theimer-Lienhard, X., and Vernikos, G. (2023). GPoeT: a Language Model Trained for Rhyme Generation on Synthetic Data.
- Prince, A., and Smolensky, P. (1997). Optimality: From neural networks to universal grammar. *Science*, 275(5306):1604–1610.

## A Complete Few Shot Prompt

Continue the poem with the next rhyming line. Ensure that the last word of the generated line rhymes with the last word of the provided line. Here are some example couplets.

Couplet 1: In the whispering calm of the starlit light  
Guided by the moon's soft, ethereal night

Couplet 2: Autumn's dance as the golden leaves release  
Preparing the ground for nature's grand feast

Couplet 3: We searched the forest, every nook and every spot.

Through thickets and brambles, our path was cut

## B GPT-3.5 Turbo Performance with Simple Prompt

First Line	Next Line Corpus	Next Line GPT 3.5 (simple)	Type of Example
his ship or by mischance	and minish his substance	he stumbled upon a mysterious island's expanse	Both Rhyme
but then he thought	that way was nought	about all the things he had been taught	Both Rhyme
and by and by	covetously	we'll reach the sky	GPT Rhymes
for evry whit	by his own wit	of darkness, there's a beam of light	Corpus Rhymes
that certainly	his policy	caught my attention	Corpus Rhymes

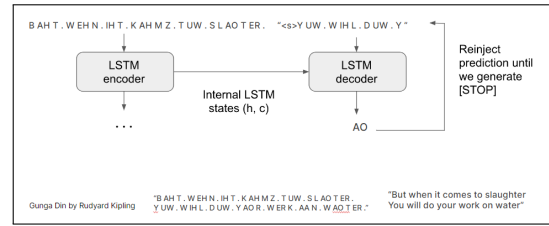
## C GPT-3.5 Turbo Performance with Descriptive Prompt

First Line	Next Line Corpus	Next Line GPT 3.5 (descriptive)	Type of Example
for to begin withal	that money was too small	we embark on a journey, big or small	Both Rhyme
hath with good cast	be rich at last	A future bright, it shall not be surpassed	Both Rhyme
he made a good purvey	and took another way	of all things wild and furry	GPT Rhymes
he diggd it in a pot	and there he left it not	and stirred it with a spoon	Corpus Rhymes
but he doubtless	of his meekness	knew how to hustle	Corpus Rhymes

## D GPT-3.5 Turbo Performance with Few Shot Prompt

First Line	Next Line Corpus	Next Line GPT 3.5 (few-shot)	Type of Example
his money to employ	to see it was a joy	But he soon realized it brought him no joy	Both Rhyme
men with some wile	might him beguile	Can charm a stranger with just a smile	Both Rhyme
yet ere this day	i have heard say	is through, We shall conquer all obstacles that come into view.	Corpus Rhymes
and by and by	covetously	we reached the water's edge so wide and vast	Corpus Rhymes

## E Sequence-to-Sequence Model with Phoneme Representation



## F ARPAbet Phoneme Table

Phoneme	Example	Translation
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER