DIMSIM: An Accurate Chinese Phonetic Similarity Algorithm based on Learned High Dimensional Encoding

Min Li
IBM Research
minli@us.ibm.com

Marina Danilevsky
IBM Research
mdanile@us.ibm.com

Sara Noeman
IBM
noemans@eg.ibm.com

Yunyao Li
IBM Research
yunyaoli@us.ibm.com

Abstract

Phonetic similarity algorithms identify words and phrases with similar pronunciation which are used in many natural language processing tasks. However, existing approaches are designed mainly for Indo-European languages and fail to capture the unique properties of Chinese pronunciation. In this paper, we propose a high dimensional encoded phonetic similarity algorithm for Chinese, DIMSIM. The encodings are learned from annotated data to separately map initial and final phonemes into n-dimensional coordinates. Pinyin phonetic similarities are then calculated by aggregating the similarities of initial, final and tone. DIMSIM demonstrates a $7.5X$ improvement on mean reciprocal rank over the state-of-the-art phonetic similarity approaches.

1 Introduction

Performing the mental gymnastics of transforming 'I'm hear' to 'I'm *here*,' or, 'I can't so buttons' to 'I can't *sew* buttons,' is familiar to anyone who has encountered autocorrected text messages, punny social media posts, or just friends with bad grammar. Although at first glance it may seem that phonetic similarity can only be quantified for audible words, this problem is often present in purely textual spaces, such as social media posts or text messages. Incorrect homophones and synophones, whether used in error or in jest, pose challenges for a wide range of NLP tasks, such as named entity identification, text normalization and spelling correction (Chung et al., 2011; Xia et al., 2006; Toutanova and Moore, 2002; Twiefel et al., 2014; Lee et al., 2013; Kessler, 2005). These tasks must therefore successfully transform incorrect words or phrases ('hear','so') to their phonetically similar correct counterparts ('here','sew'), which in turn requires a robust representation of phonetic similarity between word pairs. A reliable approach for generating phonetically similar words is equally crucial for Chinese text (Xia et al., 2006).

Pinyin	initial	final	tone
xi1	x	i	1
fan4	f	an	4

Table 1: Example Pinyins.

偶(ou2,我wo2)稀饭(xi1fan4,喜欢xi2huan1)你。 I like you.
杯具(bei1ju4,悲剧bei1ju4)啊，为一个女孩纸(zhi2,子zi5)这么香菇(xiang1gu1,想哭 xiang2ku1)。 Sadly, I am heart broken for a girl.

Table 2: Microblogs using phonetic transcription.

Unfortunately, most existing phonetic similarity algorithms such as Soundex (Archives and Administration, 2007) and Double Metaphone (DM) Philips (2000) are motivated by English and designed for Indo-European languages. Words are encoded to approximate phonetic presentations by ignoring vowels (except foremost ones), which is appropriate where phonetic transcription consists of a sequence of phonemes, such as for English. In contrast, the speech sound of a Chinese character is represented by a single syllable in Pinyin consisting of two or three parts: an *initial* (optional), a *final* or *compound finals*, and *tone* [1] (Table 1). As a result, phonetic similarity approaches designed for Indo-European languages often fall short when applied to Chinese text. Note that we use Pinyin as the phonetic representation because it is a widely accepted Romanization system (San, 2007; ISO, 2015) of Chinese syllables, used to teach pronunciation of standard Chinese. Table 2 shows two sentences from Chinese microblogs, containing informal words derived from phonetic transcription. The DM and Soundex encodings for

[1] Chinese has five tones, represented on a 1-5 scale.

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 444–453
Brussels, Belgium, October 31 - November 1, 2018.

Words	DM	Soundex
稀xi1饭fan4	S:S,FN:FN	X000,F500
喜xi2欢huan1	S:S,HN:HN	X000,H500
泄xie4愤fen4	S:S,FN:FN	X000,F500

Table 3: DM and Soundex of Chinese words.

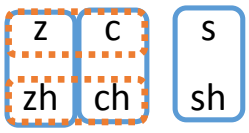

Figure 1: Grouping initials by phonetic similarity.

near-homonyms of 喜欢 from Table 2 are shown in Table 3. Since both DM and Soundex ignore vowels and tones, words with dissimilar pronunciations are incorrectly assigned to the same encoding (e.g. 稀饭 and 泄愤), while true near-homonyms are encoded much further apart (e.g. 稀饭 and 喜欢). On the other hand, additional candidates with similar phonetic distances such as 心xin1烦fan2，西xi1方fang1 for 稀饭 should be generated, for consumption by downstream applications such as text normalization.

The example highlights the importance of considering all Pinyin components and their characteristics when calculating Chinese phonetic similarity (Xia et al., 2006). One recent work (Yao, 2015) manually assigns a single numerical number to encode and derive phonetic similarity. However, this single-encoding approach is inaccurate since the phonetic distances between Pinyins are not captured well in a one dimensional space. Figure 1 illustrates the similarities between a subset of initials. Initial groups "z, c", "zh, ch", "z, zh" and "zh, ch" are all similar, which cannot be captured using a one dimensional representation (e.g., an encoding of "zh=0,z=1,c=2,ch=3" fails to identify the "zh, ch" pair as similar.) ALINE (Kondrak, 2003) is another illustration of the challenge of manually assigning numerical values in order to accurately represent the complex relative phonetic similarity relationships across various languages. Therefore, given the perceptual nature of the problem of phonetic similarity, it is critical to learn the distances based on as much empirical data as possible (Kessler, 2005), rather than using a manually encoded metric.

This paper presents DIMSIM, a learned n-dimensional phonetic encoding for Chinese along with a phonetic similarity algorithm, which uses the encoding to generate and rank phonetically similar words. To address the complexity of relative phonetic similarities in Pinyin components, we propose a supervised learning approach to learn n dimensional encodings for finals and initials where n can be easily extended from one to two or higher dimensions. The learning model derives accurate encodings by jointly considering Pinyin linguistic characteristics, such as place of articulation and pronunciation methods, as well as high quality annotated training data sets. We compare DIMSIM to Double Metaphone(DM), Minimum edit distance(MED) and ALINE demonstrating that DIMSIM outperforms these algorithms by $7.5X$ on mean reciprocal rank, $1.4X$ on precision and $1.5X$ on recall on a real-world dataset. Our contributions are:

1. An encoding for Chinese Pinyin leveraging Chinese pronunciation characteristics.
2. A simple and effective phonetic similarity algorithm to generate and rank phonetically similar Chinese words.
3. An implementation and a comprehensive evaluation showing the effectiveness of DIMSIM over the state-of-the-art algorithms.
4. A package release of the implemented algorithm and a constructed dataset of Chinese words with phonetic corrections.[2]

2 Generating Phonetic Candidates

DIMSIM generates ranked candidate words with similar pronunciation to a seed word. Similarity is measured by a phonetic distance metric based on n-dimensional encodings, as introduced below.

2.1 Phonetic Comparison for Pinyin

An important characteristic of Pinyin is that the three components, initial, final and tone, can be independently phonetically compared. For example, the phonetic similarity of the finals "ie" and "ue" is identical in the Pinyin pairs {"$xie2$","$xue2$"} and {"$lie2$","$lue2$"}, in spite of the varying initials. English, by contrast, does not have this characteristic. Consider as an example, the letter group "ough," which is pronounced quite differently in "rough," "through" and "though."

Note that depending on the initials, a final of same written form can represent different finals. For instance, $\ddot{u}$ is written as u after j, q and x; uo is written as o after b, p, m, f or w. There are a total

[2]https://github.com/System-T/DimSim.

of six rewritten rules in Pinyin (ISO, 2015). Since these rules are fixed, we preprocess the Pinyins according to these rules, transforming them into the original form for our internal representation (e.g., we represent *ju* as *jü* and *bo* as *buo*.)

2.2 Measuring Phonetic Similarity

DIMSIM represents a given word w as a list of characters $\{c_i|1 \leq i \leq K\}$ where K is the number of characters and p_{c_i} denotes the Pinyin of i_{th} character. The initial, final, and tone components of p_{c_i} are denoted as $p^I_{c_i}$, $p^F_{c_i}$, and $p^T_{c_i}$, respectively.

Formally, the phonetic similarity S between the pronunciation of c_i and c'_i is computed using Manhattan distance as the sum of the distances between the three pairs of components, as follows:

$$\sum_{1\leq i\leq K} S(c_i, c'_i) = \sum_{1\leq i\leq K} \{S_p(p^I_{c_i}, p^I_{c'_i}) + S_p(p^F_{c_i}, p^F_{c'_i}) + S_T(p^T_{c_i}, p^T_{c'_i})\} \quad (1)$$

Manhattan distance is an appropriate metric since the three components are independent. Any single change does not affect more than one component, and any change affecting several components is the result of multiple independent and additive changes. It follows that the similarity between two words is computed as the sum of the phonetic distances of characters. For example, the Pinyins of "童鞋" and "同学" are "*tong2xie2*" and "*tong2xue2*". The distance between "童(*tong2*)" and "同(*tong2*)" is zero; the distance between "鞋(*xie2*)" and "学(*xue2*)" is calculated as $S($ "鞋" , "学" $) = S_p(x, x) + S_p(ie, ue) + S_T(2, 2)$. Although the characters "鞋(*xie2*)" and "学(*xue2*)" are completely different, their Pinyins only differ in their finals.

2.3 Learning Pinyin Encodings

The next task is to compute encodings for initials, finals, and tones. While tonal similarity is easily handled (see Section 2.4), pairwise similarity for initials and finals is more complex. We adopt a supervised learning approach to obtain these encodings, using linguistic characteristics combined with a labeled dataset. The latter consists of word pairs, with specific pairs of initials or finals manually annotated for phonetic similarity. The set of annotated pairs between initials and finals are then used to learn the n-dimensional encodings of initials and finals, which will in turn be used for generating phonetically similar candidates.

	Labial	Alveolar	Retroflex	Alveolo-palatal	Velar
Plosive (u)	b	d			g
Plosive (a)	p	t			k
Nasal	m	n			
Affricate (u)		z	zh	j	
Affricate (a)		c	ch	q	
Fricative	f	s	sh	x	h
Liquid		l	r		
Semivowel		y and w			

(u) = unaspirated, (a) = aspirated

Figure 2: Table of Pinyin Initials.

2.3.1 Generating Similar Word Pairs

Phonetically similar word pairs are used to create annotations representing the phonetic similarity of initials, or finals. Chinese has 253 pairs of initials and 666 pairs of finals. Annotating examples of all these pairs is labor intensive and error-prone. Assuming twenty word pairs are provided as context per pair, the task quickly blows up to eighteen thousand annotations. However, we observe that the phonetic similarity of Pinyin is greatly impacted by the pronunciation methods and the place of articulation - this allows us to improve the accuracy and simplify the annotation task. Specifically, this is done by grouping Pinyin components into initial clusters and only annotating pairs within each cluster, and representative cluster pairs.

Figure 2 partitions initials into 12 clusters, consisting of "*bp*","*dt*","*gk*","*hf*","*nl*","*r*", "*jqx*", "*zcs*", "*zhchsh*", "*m*" ,"*y*" and "*w*", based on the pronunciation method and the place of articulation. "*f*" and "*h*" are grouped together as they are both fricative and sound very similar, especially for people from the southeast of China (Zhishihao, 2017). We then eliminate the comparison of pairs that are highly similar or highly dissimilar. For example, as the semivowel initials "*y*" and "*w*" are dissimilar to all other initials, we label every initial pair containing one of them with the lowest possible score. To compare between clusters, we randomly choose one initial from each cluster and generate just those comparison pairs. The number of pairs of initials decreases from 253 to 59.

We use a similar method for finals, partitioning them into six groups by the six basic vowels ("*a,o,e,i,u,ü*") (e.g., "*i,in,ing*" are clustered together.) We then use edit distance and common sequence length constraints to guide the pair generation; specifically, we compare a pair of finals if the edit distance between them is 1 or 2. Since

the length of finals on average is two, an edit distance of three means a complete change to the final, resulting in pairs with the lowest similarity. To compare finals across clusters, since the edit distance between any such pair is at least two, we compare pairs only when the length of the common sequence is at least two (for example, "ian" and "uan"), and otherwise assign the lowest possible similarity to the pairs. This drops the number of comparison pairs of finals down to 113.

After generating the comparison pairs, we create word pairs whose Pinyins only differ in the these pairs. We identify and account for several confounding factors that may affect annotation: 1) the position of the character containing the initial or final being compared; 2) the word length; and 3) the combination of initials and finals. Since most Chinese words are of length two, we only generate word pairs of length two for this task. Providing word pairs of length greater than two would not make much difference to learned encodings as long as word pairs are representative.

For a given initial (or final) pair (p_1, p_2), such as (b, p), we first generate the all possible Pinyins with a component of p_1 such as bao and $bing$. For each Pinyin py, we retrieve all the words with length two in the dictionary which also have first or second character with the same py. Example words for py="bao" include 包bao1袱fu2. For each created word w, we change the initial (or final) from p_1 to p_2, retrieve the corresponding words from the dictionary and generate the word pairs to compare. One such example is (包bao1袱fu2, 泡pao4芙fu2). Finally, from the full list we randomly select five word pairs that vary the first character, and five word pairs that vary the second character.

We invite three native Chinese speakers to perform the annotations. For each word pair, the annotators give a label on a 7 point scale representing their agreement, where the labels range from 'Completely Disagree' (1) to 'Completely Agree' (7). We calculate Krippendorff's α (Hayes and Krippendorff, 2007) for the initials and finals annotations to be 0.69 and 0.54, representing the inter-annotator agreement. For each word pair, we use Equation 2 to calculate the distance θ with the average value ϕ of labels across the annotators. Equation 2 inverts the labels so that the output can be used as a distance metric (phonetically similar initials or finals are closer together), and scales the

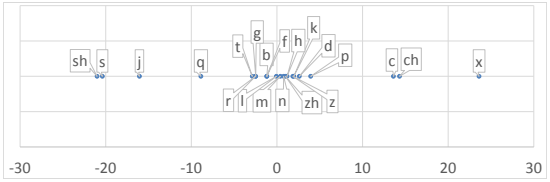

Figure 3: Learned initial encodings, n=1.

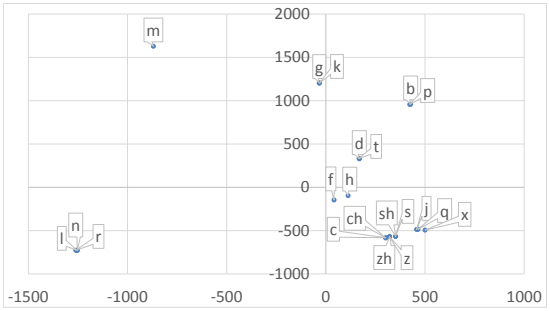

Figure 4: Learned initial encodings, n=2.

result to more accurately measure phonetic similarities. The parameters a and b are set 4 and 10^4 by default, but we also show that the performance of our method is not sensitive to the parameter settings (see Section 3.2).

$$\theta(\phi) = 1/a^{\phi} * b \tag{2}$$

2.3.2 Learning Model

Once the average distances between pairs are computed from the annotated data sets, we define a constrained optimization to compute encodings of the initials and finals. The final goal is to map each initial (or final) to an n-dimensional point.

The distance S_p of a pair p of points $(x_1, x_2, ..., x_n), (y_1, y_2, ..., y_n)$ is calculated using Euclidean distance as shown in equation 3.

$$S_p = \sqrt{\sum_{1 \le i \le n} (x_i - y_i)^2} \tag{3}$$

The model aims to minimize the sum of the absolute differences between the Euclidean distances of component pairs and the average distances obtained from the annotated training data across all pairs for initials (or finals) C. We also incorporate a penalty function, τ_p, for pairs deviating from the manually annotated distance θ so that more phonetically similar pairs are penalized more highly (we discuss τ further in Section 3.2). Equation 4 represents the cost function:

$$min \sum_{p \in C} |S_p^2 - \theta_p^2| * \tau_p \tag{4}$$

One main advantage of our learning model is that it is generic and can easily extend to any n-dimensional space. Based on the structured of Table 2, we intuit that extending beyond one dimension will yield more accurate encodings. Figures 3 and 4 visualize the computed encodings of initials when setting n=1 and n=2 We see that when $n = 2$, the locations of initial coordinates align well with Table 2,. In particular, the twelve groups are clustered in a pattern that is defined in Section 2.3.1. For example, "*bp*,*gk*,*jqx*" are separated into different clusters. However, while Table 2 indicated the basic clusters for the initials, our learned model goes further than Table 2 by actually quantifying the inter- and intra-cluster similarities. Specifically, clusters "$c, ch,\ j, q, x$" are tighter than clusters "c, c, h" and "d, t", whereas the clusters "m" and "n, r, l" are well separated from other clusters. Interestingly, the learning algorithms organically discovers new clusters that are not reflected in Table 2; namely that "r,n" and "r,l" are pairs of phonetically similar initials.

When $n = 1$, the learned model collapses the coordinates into one dimension (Figure 3). We observe that the predefined clusters are not well aligned, and many clusters are mixed together (e.g., "*bp*,*gk*,*nl*,*dt*"), preventing DIMSIM from considering variations within a cluster to be more similar than variations between clusters. Visually comparing Figures 3 and 4 gives the intuition for why DIMSIM with $n = 2$ performs better than DIMSIM with $n = 1$, which is in turn reflected in our evaluation results. Section 3 presents the effects that varying the number of dimensions has on evaluation results.

2.4 Phonetic Tone Similarity

There are five tones in Chinese, represented by a *tone number scale* ranging from 1 to 5. It is simple to use tone numbers for tone encodings and the difference between the tones of two Pinyins as the raw measure of distance, ranging in value from 1 to 5 (e.g., $S_T(xue2, xue4) = 4 - 2 = 2$). One exception is that we encode tone 3 as the numerical value of 2.5 since tone 3 is more similar to tone 2 compared to tone 4 according to the relative pitch changes of the four tones (ISO, 2015). However, this measure must first be scaled to be comparable to the pairwise phonetic distances of initials and finals. There is an additional constraint: any pairwise difference in initials or finals must have a greater negative effect on the phonetic similarity between characters than any difference in tones. For example, $S(xue1,lue1) < S(xue1,xue5)$ even though $xue1$ and $xue5$ are at opposite ends of the tone scale. We therefore scale S_T such that $Max(S_T) < Min(S_p)$.

Input : $Word\ w, Threshold\ th, Dict\ dict$;
Output: $Words\ outws$;
begin
  $pys =$ `getPinyins(`*w,dict*`)`;
  $headPys =$ `getSimPinyins(`$pys(0), th$`)`;
  $headWords =$ `getWordswithHeadPy(`$headPys, dict$`)`;
  for $cw \in headWords$ **do**
    if $cw.size \neq w.size$ **then**
      $continue$;
    end
    $sim =$ `getSimilarity(`*cw,w*`)`;
    if $sim \leq th$ **then**
      $outws.add(cw)$;
    end
  end
  `sortByAscSim(`*outws*`)`;
  $return\ outws$;
end
Algorithm 1: Generating phonetic candidates.

2.5 Candidate Generation and Ranking

Having determined the phonetic encodings and the mechanism to compute the phonetic similarity using learned phonetic encodings, we now describe how to generate and rank similar candidates in Algorithm 1. Given a word w, a similarity threshold th, and a Chinese Pinyin dictionary $dict$, we retrieve the Pinyin py of w from $dict$. We derive a list of Pinyins Pys whose similarity to py falls within the threshold th. These are used to generate a list of words with the same Pinyin in Pys and the same number of characters as w. We calculate the similarity of each candidate word with w using Equation 1 and filter out candidates that fall outside the similarity threshold th. Thus, th is a parameter that affects the precision and recall of the generated candidates. A larger th generates more candidates, increasing recall while decreasing precision.[3] Finally, we output the candidates ranked in ascending order by similarity distance.

[3]We study the impact of varying th in Section 3.

3 Evaluation

We collect 350 words from social media (Wu, 2016), and annotate each with 1-3 phonetically similar words. We use a community-maintained free dictionary to map characters to Pinyins (CEDict, 2016). We compare DIMSIM with Double Metaphone (DM) (Philips, 2000), ALINE (Kondrak, 2003) and Minimum edit distance (MED) (Navarro, 2001) in terms of precision (P), recall (R), and average Mean Reciprocal Rank (MRR) (Voorhees and et al., 1999). We calculate recall automatically using the the full test set of word pairs (Wu, 2016). Since downstream applications will only consider a limited number of candidates in practice, we evaluate precision via a manual annotation task on the top-ranked candidates generated by each approach. DM considers word spelling, pronunciation and other miscellaneous characteristics to encode the word into a primary and a secondary code. DM as one of the baselines is known to perform poorly at ranking the candidates (Carstensen, 2005) since only two codes are used. We therefore use our method (Equation 1) to rank the DM-generated candidates, to create a second baseline, DM-rank.[4] The third baseline, ALINE, measures phonetic similarity based on manually coded multi-valued articulatory features weighted by their relative importance with respect to feature salience (again, manually determined). MED, the last baseline, computes similarity as the minimum-weight series of edit operations that transforms one sound component into another.

3.1 The Effectiveness of DIMSIM

Recall and MRR: We compare DIMSIM to DM, DM-rank, ALINE and MED. DIMSIM1 and DIMSIM2 denotes DIMSIM encoding dimension $n = 1$ and $n = 2$, respectively. As shown in Figure 5, DIMSIM2 improves recall by factors of 1.5, 1.5, 1.3 and 1.2, and improves MRR by factors of 7.5, 1.4, 1.03 and 1.2 over DM, DM-Rank, ALINE and MED, respectively. DM performs relatively poorly, as it is designed for English, and does not accurately reflect Chinese pronunciation. Ranking DM candidates using the DIMSIM phonetic distance defined in Equation 1 improves its average MRR by a factor of 5.5. However, even DM-Rank is outperformed by the simple MED

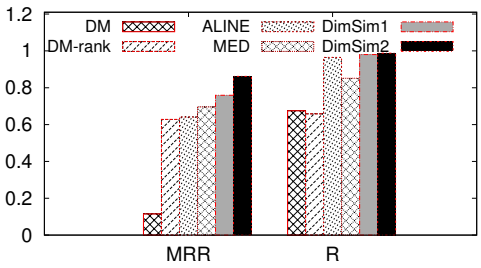

Figure 5: Recall and MRR.

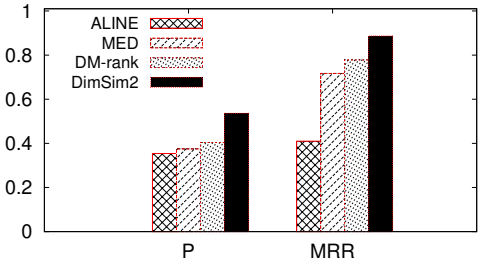

Figure 6: Precision and MRR.

baseline, demonstrating the inherent problem with DM's coarse encodings. While ALINE has a similar recall to DIMSIM, it performs worse on MRR than DIMSIM2 because it does not have a direct representation of compound vowels for Pinyin. It measures distance between compound vowels using phonetic features of basic vowels which leads to inaccuracy. In turn, MED struggles with representing accurate phonetic distances between initials, since most initials are of length 1, and the edit distance between any two characters of length 1 is identical. In contrast, DIMSIM encodes initials and finals separately, and thus even a 1-dimensional encoding (DIMSIM1) outperforms the other baselines. Finally, the intuition of Figures 3 and 4 is reflected in the data, as DIMSIM2 outperforms DIMSIM1 by 14% (MRR).

Precision and MRR: Here we evaluate the quality of the candidate ranking since in practice, downstream applications consider only a small number of possible candidates for every word. We ask two native Chinese speakers to annotate the quality of the generated candidates. Choosing 100 words randomly from the test set, we use DM-rank, MED, ALINE and DIMSIM2 to generate top-K candidates for each seed word ($K = 5$).[5] The annotators mark each candidate as phonetically similar to the seed word (1) or not (0), also marking the one candidate they believe to be the most similar-sounding (2), which may be any of

[4] We do not compare with Soundex as DM is accepted to be an improved phonetic similarity algorithm over Soundex.

[5] We do not evaluate DM and DIMSIM1 as they perform worse than DM-Rank and DIMSIM2, respectively.

the top-K candidates. We then compute precision and average MRR using the obtained annotations. We achieve inter-level agreement(ILA) of 0.75 for P and ILA of 0.84 for average MRR. DIMSIM once again outperforms MED and DM-Rank by up to $1.4X$ for precision and $1.24X$ for MRR. Since the only criteria for picking the best top-K candidate is phonetic similarity, this demonstrates that DIMSIM ranks the most phonetically similar candidates higher than the other baselines.

$\tau(\phi)$ \ $\theta(\phi)$	$1/2^{\phi}$ $*10^2$	$1/4^{\phi}$ $*10^4$	$1/\phi^2$ $*10^2$	$1/\phi^4$ $*10^3$
None	F10	F20	F30	F40
2^{ϕ}	F11	F21	-	-
4^{ϕ}	F12	F22	-	-
ϕ^2	-	-	F33	F34
ϕ^4	-	-	F43	F44

Table 4: Variations of θ,τ.

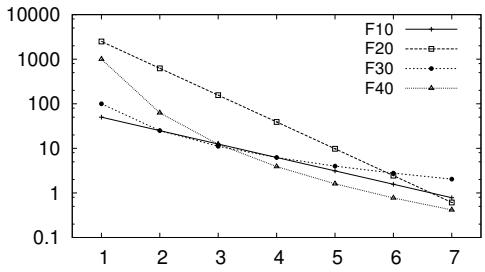

Figure 7: Distance by θ.

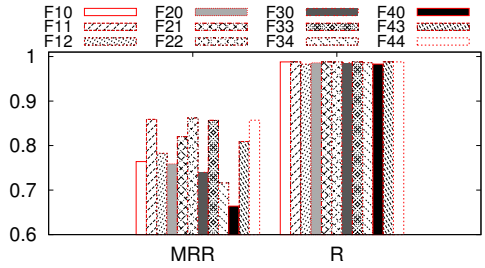

Figure 8: Impact of varying θ,τ.

3.2 Impact of Scoring and Penalty Functions

We study the sensitivity of DIMSIM to varying the scoring and penalty functions, using recall and average MRR for evaluation. Table 4 shows four different scoring functions θ and penalty functions τ (including the variation of not using a penalty function) to convert the annotator scores ϕ to pairwise distances S, following Equation 4.

Figure 7 depicts the values of the four scoring functions θ as a function of the annotator scores on a log 10 scale, to demonstrate the effect of varying a and b, as well as using ϕ as the base or exponent. Figure 8 demonstrates how sensitive our model is to the different combinations of scoring and penalty functions. We see that although Recall is entirely insensitive to the variations, the performance of MRR is impacted. There is a clear preference for the variations on the "diagonal" of Table 4: F11, F22, F33, F44, but the near-identical performance of these variations demonstrates DIMSIM's robustness to the particular scoring and penalty functions used. Note that not using a penalty function impacts MRR significantly.

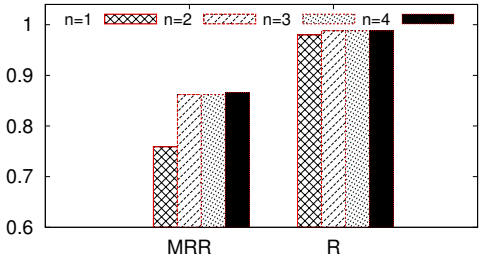

Figure 9: Impact of varying n.

3.3 Impact of the Encoding Dimensions

As demonstrated above, encoding initials and finals into a two-dimensional space is more effective than a one-dimensional space. Figure 9 presents the results of continuing to increase the number of dimensions, $n = [1, 4]$. We observe that recall is barely affected, with all variations able to successfully identify the targeted words 98% to 99% of the time. We also see that moving from n=1 to n=2 increases the average MRR by $1.14X$. However, further increasing the number of dimensions to n>2 no longer improves average MRR, indicating that learning a two-dimensional encoding is enough to capture the phonetic relationships between Pinyin components.

3.4 Impact of the Distance Threshold

We examine how the similarity distance threshold (th) impacts DIMSIM by varying th from 2 to 4096 (Figure 10) (using the scoring function F22). As th increases, recall increases from 0.75 to 0.99, converging when th reaches 2048. By increasing th DIMSIM matches more characters that are simi-

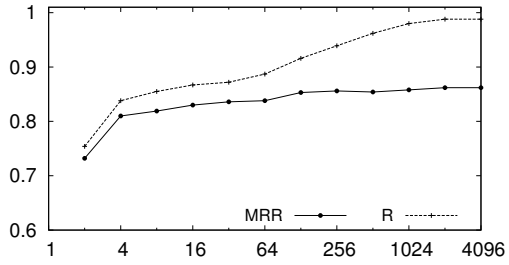

Figure 10: Impact of varying th.

lar to the first character of the given word, which in turn increases the number of candidates within the distance. Thus, the probability of including the labeled gold standard words in the results increases. MMR is less sensitive to th, converging when th reaches 128. However, the generated set of candidate words is reduced too much for $th < 128$, hurting the performance of MMR. To ensure both high recall and MRR we set $th = 2500$.

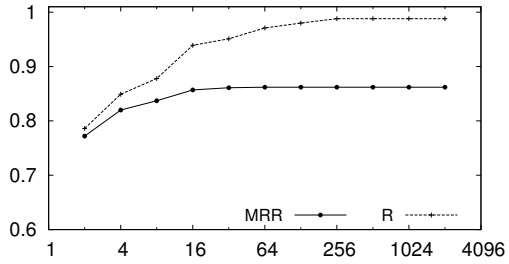

Figure 11: Varying candidate n_c.

3.5 Impact of Number of Candidates

While generating more candidates improves the recall, presenting too many candidates to a downstream application is not desirable. To find a balance, we study the impact of varying the upper limit of the number of generated candidates n_c from 2 to 2048 (Figure 11). We find that MRR converges at 64 candidates, while recall takes longer; however, setting the upper limit at 64 candidates already achieves almost 98% recall, suggesting it as a reasonable cutoff in practice. Unless otherwise mentioned, we set $n_c = 1,000$ for experiments, to isolate the impact of this parameter.

3.6 Error Analysis

We analyze and summarize three types of errors made by DIMSIM. The first occurs when targeted words are out of vocabulary(OOV). For instance, for the original word "药丸" , the targed word is "要完" which is OOV. As is commonly the case in text normalization applications which convert informal language to well-formed terms, our method works as long as the targeted words are in the dictionary. This shortcoming is generally alleviated by adding new terms to the dictionary. Second, DIMSIM cannot derive phonetic candidates from dialects that are not encoded in our mapping table. For example, for "冻($dong4$)蒜($suan4$)" , the targeted word "当($dang1$)选($xuan2$)" is obtained using the pronunciation of southern Fujian dialect. However, our approach can easily be extended to incorporate and capture such variants by learning mapping tables for each dialect and using them to generate corresponding candidates. Finally, we constrain DIMSIM to not identify candidates that differ in length from the seed word, as we observe that most transcriptions have the same word length - though some corner cases do occur.

4 Related Work

There is a plethora of work focusing on the phonetic similarities between words and characters (Archives and Administration, 2007; Mokotoff, 1997; Taft, 1970; Philips, 1990, 2000; Elsner et al., 2013). These algorithms encode words with similar pronunciation into the same code. For example, Soundex (Archives and Administration, 2007) converts words into fixed length code through a mapping table of initial groups to ordinal numbers. These algorithms fail to capture Chinese phonetic similarity since the conversion rules do not consider pronunciation properties of Pinyin. Linguists in the phonetic and phonology community have also proposed several phonetic comparison algorithms (Kessler, 2005; Mak and Barnard, 1996; Nerbonne and Heeringa, 1997; Ladefoged, 1969; Kondrak, 2003) for determining the similarity between speech forms. However, as features of articulatory phonetics are manually assigned, these algorithms fall short in capturing the perceptual essence of phonetic similarity through empirical data (Kessler, 2005). In contrast, DIMSIM achieves high accuracy by learning the encodings both from high quality training data sets and linguistic Pinyin features.

Several works in Named Entity translation (Lin and Chen, 2002; Lam et al., 2004; Kuo et al., 2007; Chung et al., 2011) focus on learning the phonetic similarity between English and Chinese automatically. These approaches first represent English and

Chinese words in basic phoneme units and apply edit distance algorithms to compute the similarity. Training frameworks are then used to learn the similarity. However, the phonetic similarity used in these systems cannot be applied to Chinese words since Pinyin has its own specific characteristics, which do not easily map to English, for determining phonetic similarity. Another main application of phonetic similarity algorithms is text normalization (Xia et al., 2006; Li et al., 2003; Han et al., 2012; Sonmez and Ozgur, 2014; Qian et al., 2015), where phonetic similarity is measured by a combination of initial and final similarities. However, the encodings used in these approaches are too coarse-grained, yielding low F1 measures. DIMSIM learns separate high dimensional encodings for initials and finals, and uses them to calculate and rank the distances between Pinyin representations of Chinese word pairs. Karl Stratos (Stratos, 2017) proposes a sub-character architecture to deal with the data sparsity problem in Korean language processing by breaking down each Korean character into a small set of primitive phonetic units. However, this work does not address the problem of the phonetic similarity and is thus orthogonal to DIMSIM.

5 Conclusion

Motivated by phonetic transcription as a widely observed phenomenon in Chinese social media and informal language, we have designed an accurate phonetic similarity algorithm. DIMSIM generates phonetically similar candidate words based on learned encodings that capture the pronunciation characteristics of Pinyin initial, final, and tone components. Using a real world dataset, we demonstrate that DIMSIM effectively improves MRR by $7.5X$, recall by $1.5X$ and precision by $1.4X$ over existing approaches.

The original motivation for this work was to improve the quality of downstream NLP tasks, such as named entity identification, text normalization and spelling correction. These tasks all share a dependency on reliable phonetic similarity as an intermediate step, especially for languages such as Chinese where incorrect homophones and synophones abound. We therefore plan to extend this line of work by applying DIMSIM to downstream applications, such as text normalization.

References

National Archives and Records Administration. 2007. The Soundex Indexing System. https://www.archives.gov/research/census/soundex.html.

Adam Carstensen. 2005. An Introduction to Double Metaphone and the Principles Behind Soundex. http://www.b-eye-network.com/view/1596.

CEDict. 2016. CC-CEDICT. https://www.mdbg.net/chindict/chindict.php?page=cc-cedict.

Jen-Ming Chung, Fu-Yuan Hsu, Cheng-Yu Lu, Hahn-Ming Lee, and Jan-Ming Ho. 2011. Automatic english-chinese name translation by using web-mining and phonetic similarity. In *IEEE International Conference on Information Reuse and Integration*, pages 283–287.

Micha Elsner, Sharon Goldwater, Naomi Feldman, and Frank Wood. 2013. A joint learning model of word segmentation, lexical acquisition, and phonetic variability. In *Proc. EMNLP*.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proc. of the joint conference on EMNLP and CoNLL*, pages 421–432. ACL.

Andrew F. Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, 1(1):77–89.

ISO. 2015. ISO 7098: Romanization of Chinese. https://www.iso.org/standard/61420.html.

Brett Kessler. 2005. Phonetic comparison algorithms. *Transactions of the Philological Society*, 103(2):243–260.

Grzegorz Kondrak. 2003. Phonetic alignment and similarity. *Computers and the Humanities*, 37(3):273–291.

Jin-Shea Kuo, Haizhou Li, and Ying-Kuei Yang. 2007. A phonetic similarity model for automatic extraction of transliteration pairs. *ACM Transactions on Asian Language Information Processing (TALIP)*, 6(2):6.

Peter Ladefoged. 1969. The measurement of phonetic similarity. In *Proceedings of the 1969 conference on Computational linguistics*, pages 1–14. Association for Computational Linguistics.

Wai Lam, Ruizhang Huang, and Pik-Shan Cheung. 2004. Learning phonetic similarity for matching named entity translations and mining new translations. In *Proc. ACM SIGIR*, pages 289–296.

Chia-ying Lee, Yu Zhang, and James R Glass. 2013. Joint learning of phonetic units and word pronunciations for asr. In *EMNLp*, pages 182–192.

Hong-lian Li, Wei He, and Bao-zong Yuan. 2003. An kind of chinese text strings' similarity and its application in speech recognition. *Journal of Chinese Information Processing*, 17(1):60–64.

Wei-Hao Lin and Hsin-Hsi Chen. 2002. Backward machine transliteration by learning phonetic similarity. In *Proc. CoNLL*, pages 1–7. ACL.

Brian Mak and Etienne Barnard. 1996. Phone clustering using the bhattacharyya distance. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 4, pages 2005–2008. IEEE.

Gary Mokotoff. 1997. Soundexing and genealogy. *Avotaynu*.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

John Nerbonne and Wilbert Heeringa. 1997. Measuring dialect distance phonetically. In *Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON-97)*.

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language*, 7(2):6.

Lawrence Philips. 2000. The double metaphone search algorithm. *C/C++ users journal*, 18(6):38–43.

Tao Qian, Yue Zhang, Meishan Zhang, Yafeng Ren, and Dong-Hong Ji. 2015. A transition-based model for joint segmentation, pos-tagging and normalization. In *EMNLP*, pages 1837–1846.

Duanmu San. 2007. *The Phonology of Standard Chinese*. Oxford University Press.

Cagil Sonmez and Arzucan Ozgur. 2014. A graph-based approach for contextual text normalization. In *EMNLP*, pages 313–324.

Karl Stratos. 2017. A sub-character architecture for korean language processing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 721–726.

Robert L Taft. 1970. *Name search techniques*. 1. Bureau of Systems Development, New York State Identification and Intelligence System.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proc. ACL*, ACL '02, pages 144–151, Stroudsburg, PA, USA.

Johannes Twiefel, Timo Baumann, Stefan Heinrich, and Stefan Wermter. 2014. Improving domain-independent cloud-based speech recognition with domain-dependent phonetic post-processing. In *AAAI*, pages 1529–1536.

Ellen M Voorhees and et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82.

You Wu. 2016. Commonly used phonetic vocabulary. http://www.51wendang.com/doc/97585e99067d692a1bbaec92.

Yunqing Xia, Kam-Fai Wong, and Wenjie Li. 2006. A phonetic-based approach to chinese chat text normalization. In *Proc. ACL*, pages 993–1000.

Mabus Yao. 2015. An algorithm for Chinese Similarity based on phonetic grapheme coding. http://mabusyao.iteye.com/blog/2267661.

Zhishihao. 2017. Differentiating f and h. http://www.zhishihao.com/xue/show/51763.