Comparative Analysis of Phonetic Algorithms Applied to Spanish

Axel A. Garcia Fuentes and Rogelio Davila

Universidad Autonoma de Guadalajara, Guadalajara, Mexico,
axel.garcia@edu.uag.mx, rogelio.davila@edu.uag.mx

Abstract. Natural language writing is a field of interest. Part of this process is related to the written language. Something that is true about that is you can make mistakes when writing or typing. Typographical errors are an example of those mistakes. Those errors may be imperceptible to a human being, but still, harmless as they may seem they may be sufficient to cause a failure when searching a record in a database . Could a phonetic algorithm capture enough of the word sound essence as to be considered a meaningful resource for colloquial Spanish?...

Keywords: Natural Language Processing, Spanish, Phonetic algorithm

1 Introduction

Recently in history there have been efforts on finding an algorithm capable of encode the sound of a word. A the beginning of the 20th century in USA those efforts were a resource during the people census in that country. Names of several places in the world had to be inputted and several of them did not have a clear writing for the data analysts. That typographic error most likely is being caused by the homophony between several names. That is called homonym[1]. Whereas Smith and Schmit sound very similar to people, certainly they are not the same for a database computer system. Something had to be done in order to properly classify or to correctly index the names of all those people.

One of the first algorithms used for this purpose is Soundex. It was used for a while until it was realized it was not helping enough for foreign names. Then some variants were used: e.g.: Russell Soundex, Daitch-Mokotoff Soundex to mention a couple of them. More phonetic algorithms were created later, e.g.: NYIIS, Metaphone, Beider-Morse Phonetic Matching, Koelner Phonetik, MRA, and PhoneticSpanish to mention some of them.

Objective of those algorithms is to create a resulting code that represents the sound that a given word may produce when it is pronounced. That means, for any arbitrary pair of words, if they share the same resulting code then they are homophones, i.e. they are pronounced exactly in the same way. The combination of the characters (letters) in the word will create the word sound when it is pronounced. In a similar way, the algorithm uses the letters that compose the word as input. Trying to mimic the effect they have in the word sound, it will use those letters interactions in order to create a sound code which will be referred as phonetic code.

Something to have in mind also is, even if we can find homophones that does not imply they are homonyms. The fact that words sound the same is not enough to cause the words to share the same meaning. There is a deep difference between sound and meaning. However, as scope definition of the discussed algorithms, for now the focus is the sound of the word. This document will disregard its meaning.

When we hear the sound of a word when it is pronounced we can remember its meaning inside our head. Maybe in the future computers could use the phonetic code for natural language processing, too. However, before getting to that stage it is necessary to chose the algorithm that best describe the sound of a word and as the document will discuss, that depends on the language. This time the discussion is about Spanish language, so that will be used for measurements and conclusions. Another, aspect to consider is the communication channel. There is oral communication and written communication. This discussion elaborates on the later leaving out the former. The question to answer is, what algorithm is the most convenient to model the sound of a word pronounced in Spanish?

2 Phonetic Algorithms

The variety of names given to the discussed algorithms so far considers: name encoding algorithms[2], phonetic algorithms[3] and phonetic coding algorithms[4], to mention some of them. The former refers to the usage these algorithms have had in recent history, notwithstanding, that name suggests those are encoding algorithms. Since what those algorithms do is tightly related to the pronunciation of the words, that name may be leaving out part of those algorithms essence. Last two names mention the concept of ”phonetics”. In linguistics there exist two major areas that study the sound of the words: i.e. phonetics and phonology. Both areas have different purposes. Understanding that is recommended to continue the discussion about these algorithms.

2.1 Phonetic and Phonology

As previously mentioned these algorithms should generate different phonetic codes depending on the sound of the word they are processing. Nevertheless, what exactly these algorithms are processing?. To answer that question it is necessary to talk a little bit about linguistics. That science has a couple of branches that study the sound of the words. They are phonetics and phonology[5]. Phonetics study the way in which pronunciation sounds are generated. For that purpose it uses acoustic analyses, as well as, analyses of what sounds are heard by the human ear. It elaborates more on the way people articulates the sounds. It considers in those analyses the parts of the phonic apparatus: tongue, throat, vocal cords, palate, soft palate, alveolus, teeth and lips[6][7][8].

Phonology is a complement. That knowledge area studies how the sounds affect the language. It studies the invariant significant elements of the words. For example, in the word "dedo" (i.e. finger in English) each one of the two letters "d" produce a slightly different sound. If that sound were switched, i.e. pronounce the second "d" as the first one and vice versa, the word would sound different (like incorrectly pronounced) but even so it would still be understood as the same word. As another example, if the second letter "d" were changed to letter "j", then the word would be "dejo" ("I leave" in English) and the meaning of the word would be different. That last is an example of a sound that should be invariant in the word "dedo"[9][10]. Linguistics call "phonemes" to those sounds that should be invariant in order to keep the word meaning. Phonology studies phonemes (among other language aspects). Phonemes vary in each language, i.e. there are different phonemes in English than there are phonemes in Spanish, which in turn are different to German, Japanese, and so on. Thus phonemes are language dependent[6][7][8].

Phonemes may be considered the functional reference to compare the sound of two words. If there is a difference in the phonemes used, then there are two different words being compared and the ideal algorithm should produce different resulting codes. Thus what the algorithm is doing is processing the word in a phonological level. That may be enough justification to name these algorithms "phonological algorithms" and to refer to their result "phonological code". However, since the name "phonetic algorithms" seems to have been widely used by now, this document will use "phonetic algorithms" and "phonetic codes" for the rest of the discussion.

2.2 Phonetic Algorithms as Hash Algorithms

Something to have in mind is phonetic algorithms were originally created to index words by the sound they produce when pronounced[11]. Using that as base, (in Spanish) "coro" and "koro" should share the same phonetic code, whereas, "coro" and "choro" should not.

Phonetic codes have been referred to as "phonetic hash". There is a number of desirable characteristics that a hash function should meet[12]. Phonetic algorithms are in deed hash functions and the reasoning is the following:

Weak Collision Resistance. For any given word, if its pronunciation significantly changes then the phonetic algorithm should generate a different phonetic code.

Strong Collision Resistance. Consider those algorithms have been designed to generate the same phonetic code for words that share the same pronunciation. In that way words can be grouped by phonetic code. Two words that sound different should have different phonetic codes. That means phonetic algorithms are designed for low collision.

Oneway Property. Phonetic algorithms will ignore characters that are irrelevant to phonetic code production. It is not possible to turn it into the initial word since there is not enough information in the phonetic code to do it.

Performance. Phonetic algorithms are deterministic. Some of them look the characters of the word in a map/dictionary that contains the phonetic code transformation. Structures like maps or dictionaries use to be O(n) where n is equals to the word length or less. Assuming that a word has a small number of characters, then their computation time is reasonable.

There are some phonetic algorithms that has been created. Some of them were designed to process text assuming an English context. Some others where extended to languages other than English. Next sections will discuss some of the phonetic algorithms that has been used in the past.

2.3 Soundex

It was used in 1930 during retrospective analysis of US census of 1890-1920[3]. It was used to index given names based on their pronunciation in English. Each Soundex code consist of a letter and three digits; e.g. W252. The letter us the initial of the surname. The digits are assigned based on the subsequent letters in the surname. The objective is to build a code of four elements; zeros are padded as required[11][13].

2.4 NYSIIS (New York State Identification and Intelligence System)

Developed circa 1970 after several attempts to improve the Soundex algorithm. NYSIIS creates a string of up to six characters. See more details in [2][13].

2.5 Daitch-Mokotoff Soundex System

Created by Ranndy Daitch and Gary Mokotoff from Jewish Genealogical Society. in 1985 Mokotoff indexed the names of 28,000 people that legally changed their names while living in Palestine during 1921 through 1948. The majority of them were Jewish with germanic or slavic names. During that exercise there were names that sounded the same. However, Soundex did not generate the same phonetic code; e.g. Moskowitz y Moskovitz[14][11][13].

2.6 Metaphone

It was published in 1990 as an enhancement to Soundex. It was designed for English language. This phonetic algorithm has been used to perform genealogical searches. There are versions of this algorithm implemented for high level languages like: Pythoh and PHP to mention some[15] [16][17][18][13]

2.7 Double-Metaphone

It was developed circa 2000. It is an improvement to Metaphone. It produces two phonetic codes for a single input word. The two phonetic codes are: the most likely pronunciation and the optimum pronunciation[2][13].

2.8 Beider-Morse Phonetic Matching

This algorithm considers that the pronunciation of a given word depends on the language where that word is being pronounced. First step is to identify the word language. Then the word is analyzed and broken down into phonetic tokens. That is performed by using pronunciation rules for the detected language. Phonetic code is the sequence of phonetic codes[19][13].

2.9 PhoneticSpanish

This is a an algorithm proposed by[20] which is based on Soundex. The algorithm tries to adapt Soundex to Spanish pronunciation. Authors does not mention any restriction on the phonetic code length.

3 Phonetic Algorithms Comparison Method

3.1 Precision Function

This section explains the shape of the mathematical function used to measure the studied phonetic algorithm.

Each phonetic algorithm generates a phonetic code for a given word: $\phi(\alpha) = \alpha'$ It is possible that a a word could be associated to a distinct phonetic code depending on the used phonetic algorithm:

$$\exists \alpha.\ \ \phi_1(\alpha) = \alpha'_1, \phi_2(\alpha) = \alpha'_2, \phi_3(\alpha) = \alpha'_3 \quad \wedge \ \ \alpha'_1 \neq \alpha'_2, \alpha'_2 \neq \alpha'_3, \alpha'_3 \neq \alpha'_1, \quad (1)$$

One of the goals of this comparison is to gather data that allows measure how the studied phonetic algorithms point to the correct word. In other words, how precise and accurate each algorithm is.

Algorithm Accuracy. Lets assume there are two words: correct word and a single evaluation word. If we were to process evaluation word to find the correct word, then that processing is one step away from the result; i.e. processing evaluation word will give as result the correct word. If we had two evaluation words the processing would be two steps away, in the worst case. If there were three evaluation words then the evaluation would be three steps away, and so on. Precision is how far the processing is from the correct word. Which in few words mean, how many words share the same phonetic code given a phonetic algorithm. Paragraphs below explain that with more details.

Algorithm Precision. The evaluation word phonetic code matches the correct word phonetic code.

The name used in this document for the set of words that share the same phonetic code is "recommendation set". Each one of those words are considered a valid replacement alternative. Consider the following premises:

1. Based on the discussion about phonetics, the pronunciation of each one of the alternatives β is equal to the pronunciation of the evaluation word α. Thus, assumption is either of those β could replace α:

$$\forall \alpha.\beta.\Theta.\{\ (p(\alpha) = p(\beta)) \rightarrow \alpha = \beta,\quad \beta \in \Theta\} \tag{2}$$

2. However, only a single one of them can be part of the solution set Ψ. The rest of them are considered false positives (FP):

$$\forall \beta.\gamma.\Theta.\Psi.\ (\beta \in \Theta \wedge \gamma \in \Theta) \wedge \gamma \neq \beta \rightarrow (\gamma \in \Psi\ \wedge\ \beta \notin \Psi\ \wedge\ \|\Psi\| = 1) \tag{3}$$

3. A False Negative (FN) is actually a True Positive (TP) incorrectly classified. Based on 3, true positives set Ψ has a single element γ. Thus, in this context FN and TP are considered the same:

$$\forall \gamma.\Psi.((\neg\neg\gamma \in \Psi \wedge \gamma \in \Psi \wedge \|\Psi\| = 1) \rightarrow \neg\neg\gamma = \gamma) \tag{4}$$

$$TP + FN = 1 \tag{5}$$

4. Phonetic codes are True Positives candidates. As 3 states there is a single one TP, the other β candidates are a False Positives(FP), and they are represented by Δ set:

$$\forall \beta.\gamma.\Delta.\ (\beta \in \Delta \wedge \gamma \notin \Delta) \rightarrow \|\Delta\| = N - 1 \tag{6}$$

Where N is the length of the alternatives set Θ.

$$\|\Theta\| = N \tag{7}$$

5. By definition phonetic codes are TP candidates. That means none of them is meant not to be an alternative candidate and thus none of them is a True Negative(TN)

$$TN = 0 \tag{8}$$

6. Accuracy equation has the following form:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{9}$$

Updating 9 with considerations of 3 and 6 precision equation has the following transformation:

$$accuracy = \frac{TP + TN}{(TP + FN) + (TN + FP)} \rightarrow \frac{TP + 0}{1 + (N - 1)} \rightarrow \frac{TP}{N} \tag{10}$$

7. Precision equation has the following form:

$$precision = \frac{TP}{TP + FP} \tag{11}$$

Updating 11 with considerations of 3 and 6 precision equation has the following transformation:

$$precision = \frac{TP}{TP + FP} \rightarrow \frac{TP}{1 + (N-1)} \rightarrow \frac{TP}{N} \tag{12}$$

From equations 10 and 12 precision and accuracy are measured in the same way. Thus, the function used for this

$$precision = \begin{cases} \frac{1}{N} & \text{Correct word found} \\ 0 & \text{Otherwise} \end{cases}$$

4 Method

A sample of 1000 tweets was used for this experiment. Those tweets were gathered without a particular constrain. The text in those tweets is natural language in Spanish spoken in Mexico. That includes slang, words created by the tweet author, text emoticons, as well as, official Spanish words. The input data also has structured strings like: urls, numbers and hashtags to mention some.

A Python script was used for this evaluation. That evaluation script uses a non-standard dictionary of terms. That Spanish dictionary of terms gathers words written in Spanish which were collected from several public resources. That made it useful for this work. Evaluation is divided in the following stages:

Phonetic Code Generation. Dictionary of terms was processed with each one of the discussed algorithms. That generated a phonetic code for each one of the terms in the dictionary. Phonetic codes were added into the dictionary of terms. The result is the phonetic codes dictionary.

Precision Data Generation. For each one of the sample tweets, each one of the words in the tweet was searched in the dictionary of terms. Some words were found and some were not. The not found words were processed with the phonetic algorithm. Their phonetic code was computed, associated to the not-found term and stored in a different data resource for later use. That means, only the not-found terms were used for the phonetic algorithm precision measurement. It was decided that way in order to have a meaningful comparison; since terms found in the dictionary of terms are correctly formed by definition.

Data generated by this processing was saved in a comma-separated values file (cvs). Each row in that file has the data generated by the processing done for a single word. In other words, if a single tweet has several not-found words then there will be multiple rows for that single tweet. The output CSV file is referred to as output map.

Results Generation. Comparison function mentioned in section 3.1 was used in this stage.

Algorithms execution time was not considered. Focus is the precision and accuracy of the studied phonetic algorithms.

4.1 Comparison Risks

Used Spanish dictionary of terms has a mixture of official and non-official Spanish words, meaning the Royal Spanish Academy(Spanish: Real Academia Española) does not recognize some of them. However, that is considered a positive aspect for this experiment because in the case of a misspelling that kind of words(non-official) are most likely to occur in natural language processing. Nevertheless, risk of a bias in the manual labeling exist. That risk is mitigated with a big sample size and re-using labeled dictionary for all of the phonetic algorithms evaluations.

5 Results

Table 1 shows the studied phonetic algorithms ascendantly ordered by mean as first criteria and then by variance. That is, the lower in the table the algorithm is the better precision it has.

Table 1. Phonetic Algorithms: Descriptive Statistics

Algorithm	**Mean**	**Mode**	**Median**	**Variance**	**Std Dev**
Phonex	0.0010	-	0.0008	0.0000	0.0006
Fuzzy Soundex	0.0081	-	0.0058	0.0000	0.0071
Rusell Index	0.0164	-	0.0115	0.0002	0.0140
Phonix	0.0169	-	0.0129	0.0002	0.0129
Alpha Sis	0.0191	-	0.0143	0.0003	0.0161
Soundex	0.0202	-	0.0165	0.0002	0.0137
Double Metaphone	0.0224	-	0.0178	0.0003	0.0179
Phonetik	0.0253	-	0.0183	0.0004	0.0207
Sfinixbis	0.0278	0.0243	0.0243	0.0003	0.0165
Caverphone	0.0317	-	0.0252	0.0006	0.0235
MRA	0.0416	0.0000	0.0374	0.0006	0.0243
Metaphone	0.0464	-	0.0416	0.0008	0.0287
BMPM	0.0532	-	0.0497	0.0009	0.0302
NYSIIS	0.0536	-	0.0474	0.0010	0.0318
Phonet	0.0838	-	0.0803	0.0013	0.0355
Phonem	0.1036	-	0.1011	0.0020	0.0444

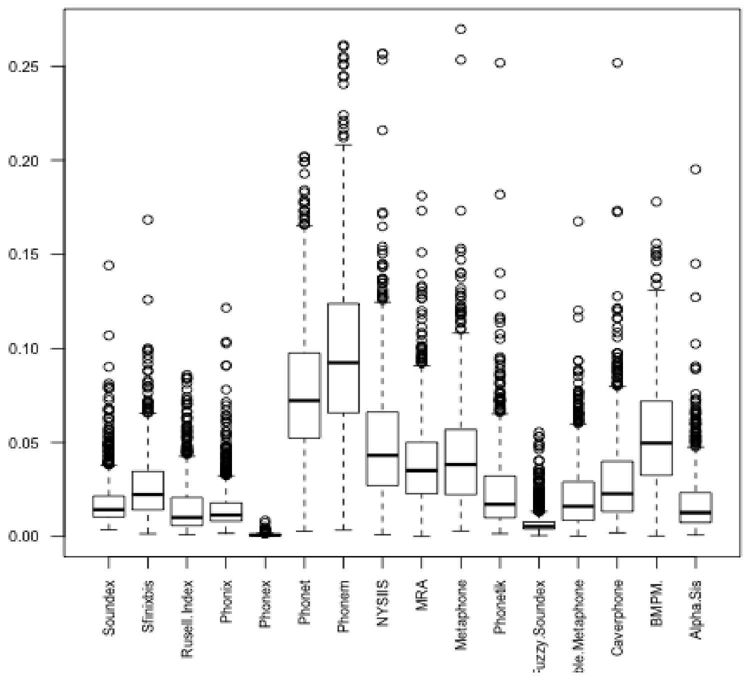

Fig. 1. default

6 Conclusions

Phonem and Phonet are the two algorithms with the best descriptive statistics since they have the highest precision mean. Phonet has lower variability than Phonem and the variance-precision ratio is better for Phonet also. That fact suggests Phonet may give better results for natural language processing applied to Spanish texts.

References

1. P. C. A. P. . C. M. C. Y. R. Castro, D. C., "Mtodos para la correccin ortogrfica automtica del espaol.,"
2. D. R. Wilson, "Name standardization for genealogical record linkage, family & church history department the church of jesus christ of latterday saints," June 2005.

3. C. Gonzales-Cam, “Algoritmos fonéticos en el desarrollo de un sistema de información de marcas y signos distintivos,” *Biblios: Revista electrónica de bibliotecología, archivología y museología*, no. 32, p. 8, 2008. [Online; accessed 20-June-2016; `http://eprints.rclis.org/12346/1/65.pdf`].
4. M. A. R. Barragn, “Similitud fontica entre palabras para mejorar la recuperacio?n de informacin en documentos orales.,” 2009.
5. X. Frías Conde, “Introducción a la fonética y fonología del español,” *Ianua. Revista Philologica Romanica*, vol. 4, p. 23, 2001.
6. uab.es, “Fontica y fonologa,”
7. “¡update title¿,”
8. F. Trujillo, “Nociones de fontica y fonologa para la prctica educativa.,” 2002.
9. Iowa, “Fontica: Los sonidos del espaol,”
10. C. R. SANCHEZ, “¡update title¿,”
11. archives.gov, “The soundex indexing system,” 2007.
12. D. P. Ning, “Csc/ece 574 computer and network security topic 4. cryptographic hash functions,”
13. python.org, “abydos 0.2.0,”
14. G. Mokotoff, “Soundexing and genealogy by gary mokotoff,”
15. searchforancestors.com, “Ancestor search metaphone calculator,”
16. python.org, “Metaphone 0.4,”
17. w3schools.com, “Php metaphone() function,”
18. python.org, “Metaphone 0.4,”
19. S. Morse, “¡update title¿,”
20. M. F. . E. J. Amn, I., “Algoritmo fontico para deteccin de cadenas de texto duplicadas en el idioma espaol.,” *Revista Ingenieras Universidad de Medelln, 11(20), 127-138.*, 2012.