Comparative Analysis of Phonetic Algorithms Applied to Spanish

Axel A. Garcia Fuentes[1], Irene Payan Parra[1],
J. Ubaldo Quevedo-Torrero[2] and Rogelio Davila Perez[1]

[1]Graduate Division in Computer Science
Universidad Autonoma de Guadalajara
Zapopan, Mexico
axel.garcia@edu.uag.mx
irene.payan.p@gmail.com
rdav90@gmail.com

[2]Computer Science Department
University of Wisconsin Parkside
Wisconsin, USA
quevedo@uwp.edu

***Abstract*—The vast and ever-increasing text posting in ”social networks” such as Facebook and Twitter, during the last 15 years, has produced an immense and rich text repository for several areas of knowledge. Therefore, text mining has recently become a very active and attractive area of research in computer science. The limited current understanding of the knowledge represented in these repositories has already influenced decision making for several companies, public figures and government agencies. However, the demand for full understanding and manipulation of these knowledge repositories has placed additional expectations on text mining. Hence, some algorithms have being developed for retrieving information, and most of them based on word recognition. This article discusses the use of phonetic algorithms to improve word recognition and proposes a precision measurement method to evaluate the results. It also presents some examples of applying some well-known phonetic algorithms on Spanish written texts using natural language processing techniques.**

***Keywords*: Natural Language Processing; text processing; phonetic algorithms; Spanish text processing**

Submission for ”Regular Research Papers”

I. Introduction

Word recognition is an important part of any natural language processing application but identifying words may be a complicated task as different combinations of letters may have the same pronunciation and it is common for people to misspell them when writing. In the USA census in 1930 [1]: having many people from different parts of the world they were confronted to capture their names. The homophony between proper names such as Smith and Schmit or Moskowitz and Moskovitz, caused typographic errors making difficult to properly classify or to correctly index those names.

One of the first algorithms used when looking for a solution to that problem is called Soundex. It was developed by Robert Russell and Margaret Odell in 1918[2], [3]. A variation of that algorithm called American Soundex was used in the US census of 1930[4]–[6]. It was used during a while until a deficiency on handling foreign names was identified [7]–[9]. After Soundex some variants were used; e.g. Daitch-Mokotoff Soundex[10], NYIIS[11], Metaphone[11], Beider-Morse Phonetic Matching[12], Koelner Phonetik, MRA, Caverphone[11] and PhoneticSpanish[13] to mention some of them. Those algorithms were called phonetic algorithms since they generate a result based on the assumed sound the letters may have when written together. More details about this type of algorithm will be discussed in the following sections.

The 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 assumed to be 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 resulting code which will be referred as phonetic code.

When analyzing words, if homophones are found, it does not imply they are homonyms, that is, have the same meaning. Two words may share the same sound but not necessarily the same meaning. There is a deep difference between sound and meaning, in this document the meaning will be disregarded. The focus of this discussion will be on how effective the existent phonetic algorithms are to represent the word being analyzed.

II. Phonetic Algorithms

There is a variety of names that have been given to the algorithms under discussion: name encoding algorithms[14], phonetic algorithms[4] and phonetic coding algorithms[15], to mention some of them. The first one evoke the usage those algorithms had during the census of 1930 in USA. Notwithstanding, they can be used to process other words besides proper names. Last two names mention the concept of ”phonetic”. In linguistics there exist two major areas that study the sound of the words: i.e. phonetics and phonology. Both areas have different purposes[16].

A. Phonetic and Phonology

In Linguistic they are two areas that study the sound of the language and the effect the sound has on the language itself. Those areas are phonetic and phonology[17]. 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[18]–[20].

According with[16] phonology studies elements called ”phonemes”. They are part of the study of how the sounds affect the language. The following example is being taken from the same authors and it is being used to illustrate what phonology studies: In a language there exist a sound set for which varying a single aspect of them causes a radical variation of its meaning. Example is the following, in the word [póka]”poca” (which in english could be interpreted as ”little” or ”few”) the first sound is a voiceless bilabial stop, [p]. If the vibration factor of the vocal cords varies when pronouncing that consonant, i.e. they are caused to vibrate, then the voiceless bilabial stop becomes a voiced bilabial stop [b]. The result of that change is a completely different word: [bóka]”boca” (mouth).

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 [18]–[20].

Phonemes may be considered the functional reference to compare the sound of two words in a given language. 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” and to refer to their result ”phonological code”.

B. 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[5]. Using that as base, (in Spanish) ”coro” and ”koro” should share the same phonetic code, whereas, ”coro” and ”choro” should not.

Phonetic codes are the result of executing phonetic algorithm on a word. They have been used to create the index of a name in the mentioned census of 1930. That resembles the usage of a hash function and in fact, phonetic codes have been informally referred to as ”phonetic hash”. There is a number of desirable characteristics that a hash function should meet[21]. Phonetic algorithms are in deed hash functions and the reasoning is the following:

- **Weak Collision Resistance.** For any two words, if their pronunciation are similar then the phonetic algorithm should generate the same 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. Hence, in most cases It is not possible from the phonetic code to return to the original word since there is not enough information in the phonetic code to accomplish 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 have complexity O(n) where n is the word length. Assuming that a word has a small number of characters, then their computation time is reasonable.

There are some phonetic algorithms that have 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.

- **Soundex.** It was used in 1930 during retrospective analysis of US census of 1890-1920[4]. The initial version of the system was patented by Margaret Odell and Robert Russell in 1918 as U.S. Pat. No. 1,261,167[3]and 1,435,663 (1922)[22]. 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. W-252. 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[5][9].
  The so-called ”American Soundex” system is an improvement on Russell's invention, and was used by the National Archives and Record Administration to index the 1880, 1890, 1900, 1910, and 1920 U.S. Censuses[3].
  With some minor changes to the weighting scheme used, Soundex has been applied to languages other than English[22][23].

TABLE I
PHONETIC ALGORITHMS: PHONETIC CODES COMPARISON

	Generated Phonetic Codes of Phonetic Algorithms	
Surname	**American Soundex**	**D-M Soundex**
Peters	P362	739400, 734000
Peterson	P362	739460, 734600
Moskowitz	M232	645740
Moskovitz	M213	645740
Auerbach	A612	097500, 097400
Uhrbach	U612	097500, 097400
Jackson	J250	154600, 454600, 145460, 445460
Jackson-Jackson	J252	154664, 454664, 145466, 445466, 154646, 454646, 145464, 445464

- **Daitch-Mokotoff Soundex System.** This variation of Soundex was Motivated by the poor results when dealing with Slavic and Yiddish surnames. Its creators were the jewish genealogist Gary Mokotoff and Ranndy Daitch 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[7][8][9].
  Table I shows some phonetic codes generated using American Soundex and D-M Soundex. That table was obtained from [24].
- **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 [14][9].
- **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 [9], [25]–[27].
- **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[14][9].
- **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[9], [28].
- **PhoneticSpanish.** This is a an algorithm proposed by[13] 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.

III. Phonetic Algorithms Comparison Method

A. 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:

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

One of the goals of this comparison is to gather data that allows to measure how the studied phonetic algorithms point to the correct word. In other words, how precise and accurate each algorithm is. The following are the definitions of precision and accuracy considered for this document:

- **Algorithm Precision.** Lets assume there are two words: ”correct word” and a single ”evaluation word”. If the evaluation word were to be processed in order to find the correct word, then that processing requires one step to produce the result; i.e. processing the evaluation word will give as result the correct word. Now assume there are two evaluation words and a single correct word. In that case, processing the result would require two steps; one for each evaluation word. If there were three evaluation words then the evaluation would be three steps, and so on. Precision is how many candidate words there exist to find 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 Accuracy.** How close the word?s resultant phonetic code matches the correct word?s 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:

$$\forall\alpha.\beta.\Theta.\{\ (\phi(\alpha) = \phi(\beta)) \rightarrow (\alpha = \beta\ \wedge\ \beta \in \Theta)\} \quad (2)$$

2) However, only a single one of them can be part of the solution set Ψ; i.e. $\|\Psi\|$=TP=1. The rest of them are considered false positives (FP):

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

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

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

$$TP + FN = 1 \quad (5)$$

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:

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

Where N is the length of the alternatives set Θ.

$$\|\Theta\| = N \quad (7)$$

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)

$$TN = 0 \quad (8)$$

6) Accuracy equation has the following form:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (9)$$

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

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

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:

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

Equations (10) and (12) have the same form. Thus a single equation can be used to measure both characteristics, i.e. precision and accuracy. However, a result that does not contain the expected value will not be considered precise:

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

IV. 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 (Mexican Spanish). 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.

It is important to mention that the language used in social networks is most likely informal. The data presents words with alterations with respect their respective formal mode. Altered words are those which do not appear in the lexical resource(i.e. Spanish dictionary). Formal words are those contained in the lexical resource.

The following are identified cases of altered words and the approach used to handle those cases:

- **Unnecessary repeated letters.** For example ”lentisssssssssimo” instead of ”lentísimo” which could be interpreted as ”extremely slow”. Another occurrence of this same example is ”Aaaaaaaaaaaah” instead of ”ah” which is an interjection used to express sorrow, exclamation, surprise or a similar feeling according with the Royal Spanish Academy. In this example both words have an exact match in the Royal Spanish Academy dictionary. For those cases, the word with unnecessary repeated letters was transformed to its Royal Spanish Academy form.

- **Incorrect Characters in a formal word.** For example ”cafŽ” instead of ”café” (coffe), ”conŽctese” instead of ”conéctese” (interpreted as ”make a connection to”), ”TŌŒcnico” instead of ”Técnico”(”Technician”). In this example both words have an exact match in the Royal Spanish Academy dictionary, too. For those cases, the word with incorrect characters has been transformed to its Royal Spanish Academy form.

- **Ambiguous Words.** For example: ”T?pico” instead of ”Tópico”(topic) or ”Típico”(typic), ”est?” instead of ”está” or ”esté” (to be). Correct term may be inferred from the context where the word is being used. However, it has been decided to ignore those words since finding the correct selection goes beyond the scope of phonetic algorithms. Doing such search may introduce systematic errors in the measurements of this experiment.

- **Ad hoc Abbreviations.** For example: ”x” instead of ”por” which could be interpret as ”for” or ”why”. Another example is ”xq” instead of ”por qué”(why) or ”porque”(because), to mention some examples. For the majority of these cases, correct term may be inferred from the context where the word is being used. However, it has been decided to ignore those words since finding the correct selection goes beyond the scope of phonetic algorithms and some of them represent more than a single word.

- **Special Characters.** This category refers to characters that appear isolated in the processed text which do not pertain to the official Spanish alphabet according with the Royal Spanish Academy[29]; e.g. @, #, -, =. Characters in this category have been ignored for the measurements.

- **Hashtags and Mentions.** Hashtags may be composed by multiple words or by aliases. Most of them will not be defined in the Royal Spanish Academy dictionary. For that reason, words in this category have been ignored.

The above described control mechanism has been used in order to prevent systematic errors on the data given that fact is to Tweet author decides to ignore orthography rules.

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 such as newspapers, Wikipedia Spanish documents, among others, what made it useful for this work.

Evaluation is divided in the following stages:

- Phonetic Code Generation. All dictionary entries were processed with each of the algorithms discussed in previous sections. Each process generated a phonetic code for each one of the terms. Phonetic codes were added into the dictionary enhancing its content and creating what could be called a ”phonetic dictionary” of Spanish terms.

- 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.

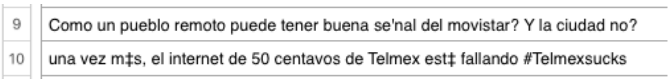
- Results Generation. Comparison function mentioned in section III-A was used in this stage.

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

V. Results

In this section, some results are presented, data samples are shown in full as the size of the results makes impractical to include the entire set. However, the full results set can be accessed in [30]

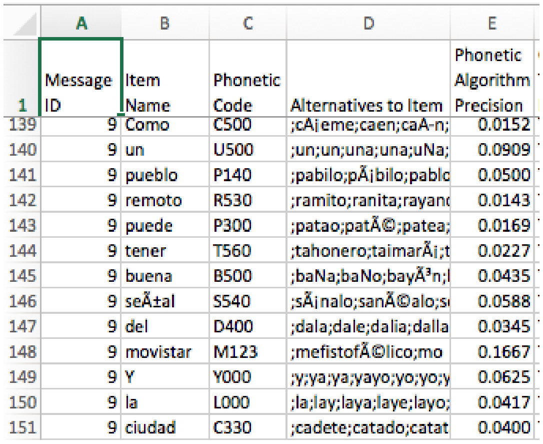
Figure 1 shows a fragment of a CSV(comma separated value) file. The figure shows a portion of the sample of 1,000 tweets in Spanish.



9	Como un pueblo remoto puede tener buena se'nal del movistar? Y la ciudad no?
10	una vez m‡s, el internet de 50 centavos de Telmex est‡ fallando #Telmexsucks

Fig. 1. Pre-processed tweet sample.

After the Python script processes the input data(see Fig. 1) it generates as output another CSV file with the result data(see Fig. 2). In the output data, the ”Message ID” column is the identifier of the words of each tweet, ”Item Name” column is the word that was analyzed, ”Phonetic Code” is the phonetic code that was assigned to the word being processed, ”Alternatives to Item” column are words that look similar to the word sound that was analyzed, ”phonetic Algorithm precision” column is the precision computed using the method described in this document.



	A	B	C	D	E
1	Message ID	Item Name	Phonetic Code	Alternatives to Item	Phonetic Algorithm Precision
139	9	Como	C500	;cÃ¡eme;caen;caA-n;	0.0152
140	9	un	U500	;un;un;una;una;uNa;	0.0909
141	9	pueblo	P140	;pabilo;pÃ¡bilo;pablo	0.0500
142	9	remoto	R530	;ramito;ranita;rayan	0.0143
143	9	puede	P300	;patao;patÃ©;patea;	0.0169
144	9	tener	T560	;tahonero;taimarÃ¡;t	0.0227
145	9	buena	B500	;baNa;baNo;bayÃ³n;l	0.0435
146	9	seÃ±al	S540	;sÃ¡nalo;sanÃ©alo;s	0.0588
147	9	del	D400	;dala;dale;dalia;dalla	0.0345
148	9	movistar	M123	;mefistofÃ©lico;mo	0.1667
149	9	Y	Y000	;y;ya;ya;yayo;yo;yo;y	0.0625
150	9	la	L000	;la;lay;laya;laye;layo;	0.0417
151	9	ciudad	C330	;cadete;catado;catat	0.0400

Fig. 2. Processed tweet sample.

At the right of Fig. 1 it is possible to see the ”Message ID”. For instance, first row shows Message ID 9. In Fig. 2, the same Message ID can be found. The resulting phonetic code of some of the words in that message are shown.

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

TABLE II
Phonetic Algorithms Precision, Ascendent by Mean

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

VI. Conclusions and Future Work

The Spanish dictionary of terms has a mixture of official and non-official Spanish words, however, that is considered a positive aspect for this experiment because in the case of a misspelling that kind of words(i.e. non-official words) are most likely to occur in natural language processing. Nevertheless, risk of a bias in the manual labeling exist. That risk has been mitigated with a big sample size and the re-usage of the same labeled dictionary for all of the phonetic algorithms evaluations.

Results show Phonem and Phonet are the two algorithms with the highest measured precision mean among the studied phonetic algorithms in this work. Although that measurement is relative to the dictionary of terms used for this experiment, it allows to compare the performance of phonetic algorithms to each other.

Studied phonetic algorithms were designed for English, Slavic and Yiddish languages. In this work they were applied to Mexican Spanish words and showed a maximum measured precision close to 10%. Future work should be done in order to find a phonetic algorithm capable of achieving a higher precision for words in Mexican Spanish.

References

[1] archives.org, “1930 federal population census.” [Online]. Available: http://www.archives.gov/research/census/1930/finding-aids.html#soundex

[2] K. Henderson, *The guru’s guide to Transact-SQL*. Addison-Wesley Longman Publishing Co., Inc., 2000.

[3] C. McPeake, H. Chen, R. E. Whalen, and D. Vezina, “System and methodology for name searches,” Apr. 16 2013, uS Patent 8,423,563.

[4] 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.

[5] archives.gov. (2007) The soundex indexing system. [Online]. Available: http://www.archives.gov/research/census/soundex.html

[6] M. Wright, “Mechanizing a large index; appendix: the soundex code,” *The Computer Journal*, vol. 3, p. 33, 1960.

[7] G. Mokotoff, “Soundexing and genealogy by gary mokotoff.” [Online]. Available: http://www.avotaynu.com/soundex.htm

[8] I. Community, “Soundex dm.” [Online]. Available: http://community.actian.com/w/index.php/Soundex_dm

[9] python.org. abydos 0.2.0. https://pypi.python.org/pypi/abydos. [Online]. Available: https://pypi.python.org/pypi/abydos

[10] JewishGen. (2016, mar) Soundex coding nara and daitch-mokotoff soundex. http://www.jewishgen.org/InfoFiles/soundex.html#DM. [Online]. Available: http://www.jewishgen.org/InfoFiles/soundex.html#DM

[11] D. Hood, "Caverphone: Phonetic matching algorithm," *Technical Paper CTP060902, University of Otago, New Zealand*, 2002.

[12] A. Beider, "Beider-morse phonetic matching: An alternative to soundex with fewer false hits," *Avotaynu: the International Review of Jewish Genealogy (Summer 2008)*, 2008.

[13] I. Amón, F. Moreno, and J. Echeverri, "Algoritmo fonético para detección de cadenas de texto duplicadas en el idioma español," *Revista Ingenierías Universidad de Medellín*, vol. 11, no. 20, pp. 127–138, 2012. [Online]. Available: http://www.scielo.org.co/pdf/rium/v11n20/v11n20a11.pdf

[14] D. R. Wilson, "Name standardization for genealogical record linkage," in *Proc of the 5th Annual family history technology workshop*. Citeseer, 2005.

[15] M. A. R. Barragán, "Similitud fonética entre palabras para mejorar la recuperación de información en documentos orales," 2009. [Online]. Available: https://ccc.inaoep.mx/~villasen/CursoSeminarioII/PropuestaAlejandroReyes.pdf

[16] A. Quilis, J. A. A. Quilis, J. A. Fernández, and A. Quilis, *Curso de fonética y fonología españolas: para estudiantes angloamericanos*. Consejo Superior de Investigaciones Científicas. Instituto Miguel de Cervantes,, 1969.

[17] X. Frías Conde, "Introducción a la fonética y fonología del español," *Ianua. Revista Philologica Romanica*, vol. 4, p. 23, 2001.

[18] J. Llisterri, "Phonetics and phonology(fonética y fonología)." [Online]. Available: http://liceu.uab.es/~joaquim/phonetics/fon_def_ambits/fonetica_fonologia.html

[19] D. Román, C. Quezada, and O. Sabaj, "Manual de introducción al estudio fonético y fonológico. pontificia universidad católica de chile, instituto de letras, departamento de ciencias del lenguaje. 2000 [1 página].[consultado 20/12/2007]." [Online]. Available: http://www.domingo-roman.net/Documentos/Manual_de_introducci_n.pdf

[20] F. Trujillo, A. González, P. Cobo, and E. Cubillas, "Nociones de fonética y fonología para la práctica educativa," 2002.

[21] D. P. Ning, "Csc/ece 574 computer and network security topic 4. cryptographic hash functions."

[22] M. G. McKenna, "Client/server database system with methods for improved soundex processing in a heterogeneous language environment," May 26 1998, uS Patent 5,758,314.

[23] Z. Bhatti, A. Waqas, I. A. Ismaili, D. N. Hakro, and W. J. Soomro, "Phonetic based soundex & shapeex algorithm for sindhi spell checker system," *arXiv preprint arXiv:1405.3033*, 2014.

[24] Wikiland. (2016, mar) Daitch?mokotoff soundex. https://www.wikiwand.com/en/Daitch%E2%80%93Mokotoff_Soundex. [Online]. Available: https://www.wikiwand.com/en/Daitch%E2%80%93Mokotoff_Soundex

[25] searchforancestors.com, "Ancestor search metaphone calculator." [Online]. Available: http://www.searchforancestors.com/utility/metaphone.php

[26] w3schools.com, "Php metaphone() function." [Online]. Available: http://www.w3schools.com/php/func_string_metaphone.asp

[27] python.org. Metaphone 0.4. https://pypi.python.org/pypi/Metaphone/0.4. [Online]. Available: https://pypi.python.org/pypi/Metaphone/0.4

[28] S. P. M. Alexander Beider. (2010, mar) Phonetic matching: A better soundex. [Online]. Available: http://stevemorse.org/phonetics/bmpm2.htm

[29] R. S. Academy. (2016) Exclusion of 'ch' and 'll' from the alphabet(exclusión de ch y ll del abecedario). http://www.rae.es/consultas/exclusion-de-ch-y-ll-del-abecedario. [Online]. Available: http://www.rae.es/consultas/exclusion-de-ch-y-ll-del-abecedario

[30] A. Garcia and I. Payan. (2016, nov) Comparative analysis phonetic algorithms spanish, github repository. https://github.com/axelgafu/articleCSCI2016/. [Online]. Available: https://github.com/axelgafu/articleCSCI2016/