On the performance of phonetic algorithms in microtext normalization

Yerai Doval[a,b], Manuel Vilares[a], Jesús Vilares[b,*]

[a] *Universidade de Vigo*
Grupo COLE, Departamento de Informática, Escola Superior de Enxeñaría Informática
Campus As Lagoas, 32004 – Ourense (Spain)
[b] *Universidade da Coruña*
Grupo LYS, Departamento de Computación, Facultade de Informática
Campus de Elviña, 15071 – A Coruña (Spain)

Abstract

User–generated content published on microblogging social networks constitutes a priceless source of information. However, microtexts usually deviate from the standard lexical and grammatical rules of the language, thus making its processing by traditional intelligent systems very difficult. As an answer, microtext normalization consists in transforming those non–standard microtexts into standard well–written texts as a preprocessing step, allowing traditional approaches to continue with their usual processing. Given the importance of phonetic phenomena in non–standard text formation, an essential element of the knowledge base of a normalizer would be the phonetic rules that encode these phenomena, which can be found in the so–called phonetic algorithms.

In this work we experiment with a wide range of phonetic algorithms for the English language. The aim of this study is to determine the best phonetic algorithms within the context of candidate generation for microtext normalization. In other words, we intend to find those algorithms that taking as input non–standard terms to be normalized allow us to obtain as output the smallest possible sets of normalization candidates which still contain the corresponding target standard words. As it will be stated, the choice of the phonetic algorithm will depend heavily on the capabilities of the candidate selection mechanism which we usually find at the end of a microtext normalization pipeline. The faster it can make the right choices among big enough sets of candidates, the more we can sacrifice on the precision of the phonetic algorithms in favour of coverage in order to increase the overall performance of the normalization system.

Keywords: microtext normalization, phonetic algorithm, fuzzy matching, Twitter, texting

*Corresponding author: tel. +34 981 167 000 ext. 1364, fax +34 981 167 160
Email addresses: yerai.doval@uvigo.es, yerai.doval@udc.es (Yerai Doval), vilares@uvigo.es (Manuel Vilares), jesus.vilares@udc.es (Jesús Vilares)

1. Introduction

With the popularization of mobile phones and Internet social networks, the use of electronic text messaging, or *texting*, has reached astonishing figures such as more than 8,000 tweets produced per second.[1] This type of communications is usually performed in real time and over platforms which impose limits on the length of the messages, as in the case of Twitter or the traditional SMS system. Because of this, the writing style of these messages clearly differs from normal standards and phenomena such as word shortenings, contractions and abbreviations are commonly used both to gain writing speed and circumvent length limitations. Thus, the original well–written *in–vocabulary–word* (IV) is replaced by an *out–of–vocabulary–word* (OOV), as in the pairs `m8`–`mate`, `bc`–`because`, or `imo`–`in my opinion`, for example.[2] In a similar way, phonetic– and graphemic–based substitutions of characters are also often abused to show a more personal or customized way of writing, as in the pairs `dawg`–`dog`, `u`–`you` or `sum`–`some`, and the pairs `5o`–`so`, `0kay`–`okay` or `th3`–`the`, respectively. Moreover, even in the case of messaging platforms where length restrictions do not actually apply (e.g. WhatsApp), it is also common to see a writing style which tries to better reflect the feelings of the writer; for example, by using character repetitions to express emphasis, as in `noooo`–`no`, `dooooo it`–`do it` or `dammmmmn`–`damn`. In general, these deviations from standard writing rules are included under the general concept of *texting phenomena* (Thurlow, 2003).

At the same time, the vast amount of data provided by social media in general and microblogging social networks in particular, constitutes an invaluable source of user–generated content of unquestionable utility for very diverse purposes: opinion mining (Vilares et al., 2017), reputation surveillance (Law et al., 2017), political analysis (Vilares et al., 2015), health surveillance (Karisani and Agichtein, 2018), crime prediction (Gerber, 2014) or disaster management (Rudra et al., 2016), to name just a few. However, the processing and analysis of such a huge amount of data is unfeasible unless intelligent automated systems are used, which are capable of dealing with this type of content by making use of Natural Language Processing (NLP) techniques and resources (Jurafsky and Martin, 2009). Unfortunately, most of the NLP tools and resources available were originally designed to deal with standard text. Consequently, texts affected by texting phenomena cannot be reliably processed using such automated tools (Gimpel et al., 2011; Ritter et al., 2011; Foster et al., 2011). As a result, two possibilities emerge in order to process this kind of texts (Eisenstein, 2013): tool adaptation, in which traditional methods which work on standard texts are reimplemented to account for the texting phenomena in non–standard texts, and text normalization.

In this work we follow the latter proposal, the so-called *(micro)text normalization*, where, given a written text affected by texting phenomena, the goal is to

[1] According to `http://www.internetlivestats.com/one-second/` on July 2018.

[2] From now on, we will use this format OOV–IV to represent those pairs formed by an OOV and its corresponding well-written IV.

obtain an equivalent text that better follows the writing rules of the corresponding standard language. A well-established approach is based on performing normalization in two phases (Han and Baldwin, 2011; Saralegi and San Vicente, 2013; Schulz et al., 2016): the first step consists of generating a set of candidate terms (i.e. possible IVs) for each input OOV; next, selection mechanisms are applied in order to rank the candidate terms and find the most likely sequence of those. Thus, the candidate generation process is crucial in this approach; there is no need to say that if the right IV does not appear between the candidates generated, it cannot be selected in the second step.

Moreover, during the candidate generation step it is common to resort to mechanisms that exploit similarities between those words in the input and those in a dictionary of the language in order to find possible candidates. Spell checkers such as the well–known `aspell` (Atkinson, 2011) implement some of these mechanisms and could be used in this scenario. Unfortunately, as can be seen in the previous texting examples, it is not uncommon for their authors to abuse the phonetic features of their language in order to shorten or customize the spelling of words. For instance, to an English speaker, the pairs `sumone`–`someone` or `u`–`you` sound the same or quite similarly, which is also the case of the emphasis example `dooooo it` when we do not take into account the prolonged /u/. Therefore, it would be highly desirable to count on a mechanism which could provide us with this kind of phonetic–based matchings, and this is precisely the purpose of the so–called *phonetic algorithms* (Odell and Russell, 1918). Given an input written word, these algorithms obtain phonetic codes that approximately indicate the way it is pronounced in a particular language. They can thus be used to match together words that differ in their written form but not that much in their pronunciation, as in the case of our previous examples.

However, contrary to expectations, a strict phonetic matching approach based on grapheme–to–phoneme transcription (Bisani and Ney, 2008) using, for example, International Phonetic Alphabet (IPA) transcriptions (Jurafsky and Martin, 2009, Ch. 7 "Phonetics"), would not be useful in this context, as its output codes would be too specific for our needs. For instance, the processing of `beat`–`bit` or `but`–`bought` would result in slightly different codes accounting for their slightly different pronunciations (/biːt/–/bɪt/ and /bʌt/–/bɔːt/, respectively),[3] thus preventing their matching. So, in the context of microtext normalization, the use of a fuzzy phonetic–based matching seems a better choice.

Surprisingly, despite the importance of such phonetic processing of microtexts (Kobus et al., 2008; Beaufort et al., 2010; Xue et al., 2011; Baldwin et al., 2013; Schulz et al., 2016) and the existence of many phonetic algorithms publicly available for the English language, we have also noticed the lack of any extensive comparison study of the performance of these algorithms for our problem domain. In this context we have decided to conduct our own study to compare the performance of several phonetic algorithms on this task. Our main objective

[3]This kind of word pairs are formally known as *minimal pairs*.

is to facilitate future developers and researchers the choice of the phonetic algorithm to be used in a particular microtext normalization setup. By improving the process of normalization candidate generation, the potential effectiveness of the ulterior selection process and, consequently, that of the whole normalization process should be notably increased. As a result, the input noise introduced into subsequent information processing systems should be greatly reduced.

Furthermore, we can model this two–step normalization procedure as an expert or intelligent system whose knowledge base specifies those transcription rules required to normalize non–standard texts. Now, if we take into consideration the importance of phonetic–based phenomena in this domain, the need for including phonetic rules in our normalization setup should be clear. In conclusion, a study such as the one presented here will be highly valuable when constructing a microtext normalization system, as it will guide the process of populating its knowledge base.

It should be noted that, although most of the research work on text normalization has been focusing on English (Baldwin et al., 2015; Han and Baldwin, 2011; Xue et al., 2011), there is also interest in applying it to other languages. In the European context, it is worth mentioning French (Kobus et al., 2008; Beaufort et al., 2010), Dutch (Schulz et al., 2016) or Spanish, the latter being a notable case thanks to the TweetNorm Workshop (Alegria et al., 2015, 2013). Additionally, other languages such as Chinese (Wang and Ng, 2013), Arabic (Duwairi et al., 2014), or even low–resource languages like Turkish (Eryiğit and Torunoğlu-Selamet, 2017) or Punjabi (Kaur and Singh, 2015) have been also receiving attention. With this in mind, the decision of using English in our experiments was mostly due to the public availability of both evaluation corpora (Baldwin et al., 2015) and ready–to–use implementations for a wide range of phonetic algorithms (see Section 3).

The structure of the rest of this article is as follows. Firstly, Section 2 introduces the reader to the phonetic algorithms to be analysed and how they work, while Section 3 deals with their implementation. In Section 4, after describing the methodology followed, the results of our experiments are presented and discussed. Next, previously related work is introduced in Section 5 and, finally, Section 6 presents our conclusions and future work.

2. Phonetic algorithms

Being able to match strings of characters which are superficially different is a key aspect of a wide range of systems such as search engines, spell checkers or, as in our case, microtext normalizers. This usually has to do with the tendency of human actors to misspell words or, in general, deviate from standard writing rules, both unintentionally and intentionally. These deviations occur due to particular similarities between words, such as their spellings differing in just one character, having the same long prefix or being pronounced in almost the same way. Thus, it would be of great interest to be able to match strings which share a particular set of features, and not (only) the totality of their constituent

characters. These types of *partial* matchings are included in what is called *fuzzy matching*.

Any regular spell checker, such as `aspell` (Atkinson, 2011), is able to perform fuzzy matching to obtain the spelling candidate corrections for an input word. In this case, those matchings are usually determined by some distance measure being lower than a given threshold value. The most widely used distance metric is the *Levenshtein distance*, commonly known as *edit distance*, which counts the number of insertions, deletions, substitutions or transpositions of characters that would need to be applied to one word to be transformed into another (Levenshtein, 1966).

A microtext normalization system operates in a similar way, although the candidate generation step also needs to account for texting phenomena, which cannot be handled by traditional edit–distance based approaches, as in the case of `nuff–enough`, `da–the` or `str8–straight`, for example. Hence, the phonetic processing of microtexts turns out to be of key importance in order to obtain meaningful sets of normalization candidates (Beaufort et al., 2010). This task can be accomplished through the use of so–called phonetic algorithms.

A *phonetic algorithm* transforms an input written word into a phonetic code which roughly indicates the way that term is pronounced in a particular language. It is important to highlight the *approximate* nature of these codes, as its purpose is to match words with *similar* pronunciations.

This work studies the most popular state–of–the–art phonetic algorithms designed for the English language in the context of the microtext normalization task.[4] It is worth noting that most of them were originally designed for the task of personal–name matching, although it is fair to assume that the phonetic phenomena initially considered would also be useful in matching other types of similarly sounding words. Consequently, it will be interesting to analyse the performance of these algorithms in the task of microtext normalization when generating normalization candidates.

Next, the different phonetic algorithms considered for this study are introduced. The most relevant features of each algorithm, their variations and the relations existing between them are detailed. Examples of their output encodings are also shown in Tables 1 and 2 for comparison. Notice that we have tried to be concise, keeping in mind that the objective of this work is not to study the algorithms themselves but to analyze their behaviour in the context of microtext normalization. Should the reader wish to go into detail about any particular algorithm, appropriate references have been included.

2.1. Soundex

Considered the first phonetic algorithm in history, the well–known and widely used *Soundex* algorithm (Odell and Russell, 1918; Odell, 1956) mainly encodes the consonants of an input word using numerical digits, but also encodes both

[4]It must be remembered that these algorithms are language–dependent.

algorithm	nuff	enough	cntrtkxn	contradiction
Soundex	N100	E520	C536	C536
Ref. Soundex	N802	E08040	C38696358	C308690603608
			0721417200000,	0721417120000,
			0621417200000,	0621417120000,
Alpha SIS	028000000000	12700000000000	0721416720000,	0721416120000,
			0621416720000	0621416120000
NYSIIS	NAF	ENAG	CNTRTC	CANTRA
Rev. NYSIIS	NAF	ENAG	CNTRTCXN	CANTRADACTAN
MRA	NF	ENGH	CNTKXN	CNTCTN
Metaphone	NF	ENKH	KNTRTKXN	KNTRTKXN
D. Metaphone	NF,NF	ANK,ANK	KNTR,KNTR	KNTR,KNTR
D–M Soundex	670000	065000	463935	463934
Caverphone 1	NF1111	ANF111	KNTTKN	KNTRTK
Caverphone 2	NF11111111	ANF1111111	KNTTKN1111	KNTRTKSN11
Beider–Morse	nuf	iinDg,iinog,iinug,inDg, inDgx,inag,inog,inogx, inug,inugx	kntrtgzn,kntrtkzn, tzntrtgzn,tzntrtkzn	kontradiktion,kontradiktn, kontradiktion,kontroditsion, kontroditsion,kontroditstion, kontradiktion,kuntraditstion, kuntrodiktion,kuntroditstion, tsontradiktion,tsontroditstion, tsontrodiktion,tsontroditstion, tsuntradiktion,tsuntraditstion, tsuntrodiktion,tsuntroditstion
F. Soundex	N1	E5	C536375	K536395
Lein	N400	E250	C213	C213
Onca	N100	E520	C536	C536
Phonex	N1	A52	C5325	C5363235
Phonix	N5,7	V5,7	C253632,285	K2536323,5
Phonix Comm	N700	v700	C536	K536
Roger-Root	02800	12700	07214	07214
StatCan	NF	ENGH	CNTR	CNTR
Eudex	648518346341351492	1556444031249426116	437444691622462482	159360686493381737 3

Table 1: Example encodings for each of the phonetic algorithms analysed (1): `nuff`–`enough` and `cntrtkxn`–`contradiction`

algorithm	da	the	onez	ones
Soundex	D000	T000	O520	O520
Ref. Soundex	D60	T60	O0805	O0803
Alpha SIS	01000000000000	010000000000000	12000000000000	120000000000000
NYSIIS	D	TH	ON	ON
Rev. NYSIIS	D	T	ON	ON
MRA	D	TH	ONZ	ONS
Metaphone	T	0	ONS	ONS
D. Metaphone	T,T	0,T	ANS,ANS	ANS,ANS
D–M Soundex	300000	300000	064000	064000
Caverphone 1	T11111	T11111	ANS111	ANS111
Caverphone 2	TA111111111	T111111111	ANS1111111	ANS1111111
Beider–Morse	da,di,do	t,ti	Ynis,Ynits,oni,oniS, onis,onits,unis	Ynis,oni,oniS, onis,unis
F. Soundex	D	T	O59	O59
Lein	D000	T000	O250	O250
Onca	D000	T000	O500	O500
Phonex	D	T	A52	A5
Phonix	D,3	T,3	V5,8	V,58
Phonix Comm	D000	T000	v800	v800
RogerRoot	01000	01000	12000	12000
StatCan	D	TH	ONZ	ONS
Eudex	8646911284551352 32	10088063165309921 28	10664523917614514324	10664523917614514196

Table 2: Example encodings for each of the phonetic algorithms analysed (2): `da`–`the` and `onez`–`ones`.

consonants and vowels in the first position using that same character. The different digits used are related to the place of articulation of the consonant (Jurafsky and Martin, 2009, Ch. 7: "Phonetics"), so the labial consonants `b`, `f`, `p` and `v` are encoded as the number `1`, for example. Before any other rule is applied, the algorithm checks for character sequences represented by the same number and chooses either to retain the first of those characters or the full sequence depending on other characters in the context. Its codes have a fixed length of four characters, padded with trailing `0`'s when needed.

Soundex conforms the basis for many other modern phonetic algorithms. These newer algorithms mostly try to address its poor precision, as in the refined version of the original algorithm (*Ref. Soundex*) available in ASF (2017), which is also tested here. This revised version does not impose a length limit on the encodings and takes vowels more into consideration for the encoding.

2.2. IBM Alpha Search Inquiring System

Popularly known as *Alpha SIS* (Moore, 1977), this algorithm uses two different conversion tables, one for the first characters of the input word and the other for the rest. The encodings are conformed by a fixed number of 14 numerical digits, appending trailing `0`'s as padding for shorter words. If two characters with the same phonetic code are adjacent, only the first one will be used, but a third character would be retained. Alpha SIS also focuses on the encoding of consonantal sounds, although vowels are encoded if appearing at the beginning of a word. It also may return multiple alternative encodings as output.

2.3. New York State Identification and Intelligence System

Commonly known as *NYSIIS* (Taft, 1970), its encoding procedure makes use of a small letter alphabet instead of numerical digits. Its more complex ruleset with respect to the Soundex algorithm allows for the processing of notable character n–grams such as `sch` or `rd`, performing different actions depending on their context characters being vowels or not, and on characters being at the end of the word. The first character of the word is maintained as–is while the rest of the vowels are replaced with the letter `a`. Finally, the code is truncated to its first six characters.

A revised version of this algorithm (*Rev. NYSIIS*), which adds new rules in order to obtain higher precision codes, is also available (Kell, 1988).

2.4. Match Rating Approach

Usually referred to as *MRA* for short, in this case we are not only talking about a phonetic algorithm but also about a particular comparison scheme for the phonetic codes (Moore, 1977). The encoding rules are quite simple: delete all vowels —except the first one if the word begins with it—, remove character repetitions and reduce the length of the code to six by using the first and last three characters only. However, the complexity of the system lies in the comparison rules. For encoded strings with a length difference less than three, a matching threshold value is first calculated using a table. Then, using a forward

and backward pass over the codes of the strings to be compared, the number of distinct characters between them is obtained. Finally, the encoded strings are considered similar depending on this number being equal or greater than the given threshold value.

2.5. Metaphone

Widely–used, the *Metaphone* algorithm is employed in the `aspell` spell checker (Philips, 1990). Its set of contextual rules maps between n–grams of characters from the source to n–grams of characters from an alphabet of sixteen consonantal symbols. Again, the main focus of this algorithm is on encoding consonantal sounds, while vowels are included only if appearing in the first position of the word. No limit on code length is imposed this time.

2.6. Double Metaphone

With respect to the original Metaphone, the Double Metaphone algorithm (*D. Metaphone*) takes into account several spelling peculiarities from different languages, including English, and outputs two alternative encodings of the input word (Philips, 2000).

Although an even newer iteration in the Metaphone family exists, named Metaphone 3, unfortunately its source code is not freely available and for this reason has not been included in this study.

2.7. Daitch-Mokotoff Soundex

The so–called Daitch–Mokotoff Soundex algorithm (*D–M Soundex*) constitutes an improvement on the original Soundex seeking to improve its precision when dealing with Slavic and Yiddish surnames (Mokotoff, 2007). The most notable differences with the original algorithm are the length of the codes, up to six characters long; the first character of the word being encoded as the rest, handling some specific character n–grams as a unit; and, lastly, the possibility of outputting multiple possible codes instead of a single one.

2.8. Caverphone

Designed by Hood (2002) to match common names from New Zealand, this algorithm (*Caverphone 1)* performs recursive substitutions and deletions on the original word following a set of rules mostly dealing with character n–grams. In its last steps, it appends a padding of `1`'s to the code to finally trim its length down to six characters, which is the fixed length for all codes.

Hood (2004) revised his algorithm later (*Caverphone 2*) by adding a few extra rules and increasing the length of the encodings to ten characters.

2.9. Beider–Morse

The main goal of the *Beider–Morse* algorithm (Beider, 2008) is to reduce the large amount of false positives usually returned by Soundex. This problem is managed by first determining the language of the input text so that a particular set of rules can be used accordingly. If the language cannot be determined, a generic set of rules is applied instead. A set of common rules is also applied after such language–specific or generic rules. These common rules account for final devoicing and regressive assimilation of consonants. Moreover, the alphabet of this system is based on the IPA (Jurafsky and Martin, 2009, Ch. 7 "Phonetics"), which is then simplified in order to merge together symbols with very similar sounds and to make it easier to write the codes using a standard keyboard. Finally, as with other improvements on the Soundex algorithm, it takes into account frequent character n–grams, code length is not limited, has a better support for vowels and outputs multiple possible encodings.

2.10. Fuzzy Soundex

This variation of the Soundex algorithm proposed by Holmes and McCabe (2002) and named Fuzzy Soundex (*F. Soundex*), employs two mapping tables in two subsequent stages. In the first, a table with mappings between character n–grams is used. Then, for the second stage, another table with mappings between individual characters and numerical digits is used to obtain the final output code, which is not restricted in length.

2.11. Lein

A simple variation of the Soundex, the *Lein* algorithm differs from the original one only in its conversion table (Lynch and Arends, 1977).

2.12. Onca

The *Onca* algorithm (Gill and Baldwin, 1987; Gill et al., 1993) merely consists of a two–step application of the NYSIIS and the Soundex algorithms previously described in Sections 2.3 and 2.1, respectively. According to its authors, the new algorithm overcomes the low precision of pure Soundex while retaining its 4–character format.

2.13. Phonex

Phonex is a Soundex–like algorithm which aims for a greater coverage of common orthographic variations (Lait and Randell, 1996). It is also influenced by the Metaphone method, hence its name. After removing trailing `s` characters, it uses two sets of rules: one for encoding leading characters and the other for the rest of the word. As in the original Soundex, 4–character codes are obtained as output.

2.14. Phonix

Not to be confused with the previous Phonex algorithm, there are two variants or interpretations of the Phonix algorithm. Its original interpretation (*Phonix*), described by Gadd (1988, 1990), specifies a set of rules to encode the input and target words as well as to obtain a matching code, called *ending-sound*, for each one of them. It then specifies another set of matching rules regarding the word encodings and ending–sounds previously obtained, which allows for the distinction between three different categories of matches: most–likely, less–likely and least–likely matches (to be referred to in this work as $Phonix_{most}$, $Phonix_{less}$ and $Phonix_{least}$, respectively). However, the common interpretation of the algorithm (*Phonix Comm*) skips the part concerning the ending–sound and operates in a similar way to the original Soundex algorithm.

2.15. Roger Root

The Soundex–like *Roger Root* algorithm (Lynch and Arends, 1977) produces 5–numerical codes using two conversion tables: one for the first characters of a word and the second one for the rest.

2.16. Census Modified Statistics Canada

Commonly known as *StatCan* for short, it is a very simple phonetic algorithm which preserves the first characters, deletes the remaining vowels and `y`'s, collapses identical adjacent characters and truncates the result to four characters (Lynch and Arends, 1977).

2.17. Eudex

The *Eudex* algorithm (Ticki, 2016) encodes words in a way that exposes the differences in their pronunciations, by calculating the Hamming distances between their codes. It returns an 8–byte array as output and makes use of four different conversion tables: two for ASCII and C1 (Latin Supplement) characters at the initial position of a word, and another two similar ones for the remaining characters. These tables were obtained using the IPA classifications of consonants and vowels, encoding the sound articulation features of each symbol into a binary format. This encoding has the property that similarly pronounced symbols correspond to codes with a small Hamming distance, but also highlighting the differences between them in the same way. For this reason, its author suggests using a similarity function that matches codes with a Hamming distance below a given threshold value.

3. Implementation of the phonetic algorithms

In order to avoid unnecessary effort and make their future use by developers and researchers easier, instead of reimplementing from scratch the specifications for the algorithms studied, we downloaded and evaluated implementations of them that are publicly available on the Internet. For each downloaded implementation, its compliance with the original specification of the corresponding

algorithm was tested. If the implementation did not comply, alternatives were sought or, when necessary, the source code was modified accordingly.[5]

After having performed this selection process, these are the open source projects from which the algorithm implementations to be used were obtained:

- Apache Commons Codec package (ASF, 2017): in the case of the implementations of the Soundex, Refined Soundex, Daitch–Mokotoff Soundex, Beider–Morse, Caverphone 1, Caverphone 2, MRA and Double Metaphone algorithms.
- `talisman` package (Plique, 2017): Alpha SIS, Eudex, Fuzzy Soundex, Lein, Onca, Phonex, Roger Root, StatCan, Metaphone and NYSIIS algorithms.
- `stringmetric` package (Madden, 2014): Refined NYSIIS algorithm.
- `phonetic_search` package (Ølsgaard, 2016): Phonix algorithm.

4. Evaluation

As stated before, this work intends to study the behaviour of the different phonetic algorithms available for English in the context of candidate generation for microtext normalization tasks. The criterion assumed for this purpose states that, for each input OOV, the best algorithms should return as output the smallest possible candidate set which still contains the corresponding equivalent normalized word.

4.1. Evaluation corpora

Seeking reproducibility, it was decided to use the two lexical normalization dictionaries which were made publicly available in the W–NUT 2015 Shared Task #2 (Baldwin et al., 2015). These dictionaries, named `utdallas` and `unimelb`, consist of text files conformed by lists of non–standard words obtained from real–world microtexts and their corresponding standard equivalents. It must be noted that non–standard words from these lists are affected by a wide range of texting phenomena, not limited to the phonetic phenomena in which this work is interested. For instance, consider the graphemic substitution in `5o` (`so`) or the abbreviation `2nd` (`second`). Despite this, the full datasets were used in the evaluation, thus resorting to error analysis to account for those instances missed by the phonetic algorithms.

In a similar way, the canonical English dictionary made available by the W–NUT 2015 organization for the task (from now on `canonical`) was used. This dictionary is the list of words from which the normalization candidates are retrieved using the phonetic algorithms. Some simple statistics of these datasets are shown in Table 3.

[5] All source code produced during this work is available at `http://www.grupocole.org/software/VCS/phon`.

dictionary	#chars/word	#entries	#OOV
`utdallas`	5	3,974	47
`unimelb`	7	41,181	96
`canonical`	8	165,458	–

Table 3: Evaluation corpora statistics. The column *#chars/word* indicates for the first two rows, corresponding to the lexical normalization dictionaries, the average number of characters per non–standard word; for the last row, corresponding to the canonical dictionary, it indicates the average per standard word. Column *#entries* indicates the number of lines in each evaluation dataset. In the case of the first two rows, *#OOV* shows the number of standard words not included in the canonical lexicon used.

4.2. Experimental methodology

In order to obtain the normalization candidates for each non–standard word of the evaluation corpora, the following procedure was applied:

1. Before the normalization process, create the phonetic lookup dictionary of key–value pairs (phonetic code, set of words) corresponding to each phonetic algorithm. Each pair groups all the words from the canonical dictionary with the same phonetic code for a particular algorithm.
2. During the normalization process, and for a given phonetic algorithm, match the phonetic codes obtained from the non–standard words of the evaluation corpora with these codes (keys) of the corresponding phonetic dictionaries. If a match is found, retrieve its corresponding set of words (values), which now constitute its normalization candidates. When multiple alternative codes are available for the same non–standard input word, the final candidate set is formed by the union of those partial sets obtained with each alternative code.

Additionally, in the case of the MRA and Phonix algorithms, their particular lookup procedures (MRA_{custom}, $Phonix_{most}$, $Phonix_{less}$ and $Phonix_{least}$, respectively), previously explained in Section 2, were also considered. Similarly, multiple results were obtained for the Eudex algorithm by varying the Hamming distance threshold value: *Eudex*, $Eudex_5$, $Eudex_{10}$ and $Eudex_{15}$ for threshold values 0 (i.e. perfect matching), 5, 10 and 15, respectively. In these cases, both the results corresponding to the general and specialized lookup procedures were obtained.[6]

Regarding the original Phonix algorithm in particular, three different sets of results were distinguished based on the likelihood of the candidates indicated by the lookup procedure: (1) one set containing the most likely candidates

[6]For the MRA and Eudex algorithms, it is worth mentioning the relatively high computational cost of their particular lookup procedures. Under their current implementations, they may be suitable for quick checks between pairs of words but not for the extensive dictionary–wide checks required in the current context.

($Phonix_{most}$); (2) one including the most and less likely candidates ($Phonix_{less}$) and (3) one which also adds the least likely candidates ($Phonix_{least}$). Moreover, results obtained using the alternative interpretation of this algorithm are also included ($Phonix_{Comm}$).

Precision, recall and F1 metrics, over a list of words from a particular evaluation dataset, are used to measure the performance of the phonetic algorithms. In the present application context, *precision* (P) is defined as the mean of the ratio of correct candidates[7] over the total number of candidates retrieved for each word, all of this over the total number of words:

$$P = \frac{\sum_{word} \frac{|hits_{word}|}{|candidates_{word}|}}{|words|} \tag{1}$$

In the case of *recall* (R), it is calculated as the number of times when the correct candidate was among the set of normalization candidates retrieved for each word over the total number of words:

$$R = \frac{|hits|}{|words|} \tag{2}$$

Finally, *F1* score is defined in the usual way by aggregating precision and recall:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{3}$$

4.3. Results and discussion

Tables 4 and 5 show the results obtained for each algorithm, sorting them by their corresponding F1 scores. As it can be seen, the scores obtained are quite low, mainly due to the low precision figures obtained in most of the cases. Accordingly, those algorithms with the highest precision scores end up at the top of this ranking, with *Eudex*, *MRA* and *Metaphone* outperforming the rest of them.

It is interesting to note that the performance of the MRA algorithm decreases ostensibly when used with its particular lookup procedure (MRA_{custom}). This procedure tries to enlarge the matching window of its otherwise high–precision codes. Thus, it is reasonable to assume that using a large canonical dictionary, as in this case, renders this procedure of little use, as it was designed to work with much smaller lists. Moreover, $Phonix_{least}$, that is, Phonix using least–likely matches, suffers from the same problem.

At this point, it should be noted that, when developing a microtext normalization system, it may be interesting to gain some recall while sacrificing some precision in exchange. This is due to the fact that a low recall tends to impose an upper limit on the overall performance of the system: there is no way of selecting the right IV unless it appears among the generated candidates.

[7] In this case, the number of correct candidates will be either 1 or 0.

algorithm	#hits	avg #cands	P	R	F1
Eudex	1,376	5.641	0.117	0.346	0.175
MRA	1,498	9.165	0.113	0.376	0.174
Metaphone	2,071	26.232	0.078	0.521	0.137
Beider–Morse	1,636	25.514	0.080	0.411	0.134
StatCan	2,225	21.415	0.070	0.559	0.124
Ref. Soundex	1,569	15.378	0.073	0.394	0.123
$Eudex_5$	1,800	94.125	0.047	0.452	0.085
Rev. NYSIIS	1,416	27.404	0.047	0.356	0.083
$Phonix_{most}$	1,681	36.503	0.044	0.422	0.080
F. Soundex	2,160	41.765	0.043	0.543	0.080
NYSIIS	1,410	17.565	0.043	0.354	0.078
Caverphone 2	2,039	69.022	0.039	0.513	0.073
Caverphone 1	2,246	98.240	0.031	0.565	0.060
$Eudex_{10}$	2,076	310.535	0.025	0.522	0.048
D–M Soundex	2,154	96.137	0.024	0.542	0.046
Alpha SIS	2,359	241.792	0.022	0.593	0.042
$Phonix_{less}$	2,191	141.575	0.020	0.551	0.040
$Eudex_{15}$	2,199	551.689	0.016	0.553	0.032
Roger Root	2,516	133.588	0.016	0.633	0.032
Soundex	2,469	80.570	0.015	0.621	0.030
D. Metaphone	2,395	84.389	0.014	0.602	0.028
Lein	2,471	101.642	0.013	0.621	0.026
Phonix Comm	2,453	194.972	0.011	0.617	0.023
Onca	2,396	109.237	0.010	0.602	0.020
Phonex	2,656	266.348	0.009	0.668	0.019
$Phonix_{least}$	2,332	611.050	0.006	0.586	0.013
MRA_{custom}	3,695	15,270.327	0.000	0.929	0.000

Table 4: Results for the `utdallas` dictionary, ranked by F1. For each phonetic algorithm, the second column (*#hits*) indicates the number of instances from the corpora for which the algorithm could provide the correct answer. The average number of normalization candidates returned for each OOV using that algorithm is shown in the third column (*avg #cands*). The rest of the columns contain the values obtained for the evaluation metrics used; from left to right: precision (P), recall (R) and F1 score (*F1*).

Moreover, it is also reasonable to assume that the candidate generation step will be connected to a capable candidate selection step afterwards. Taking this into account, Tables 6 and 7 show again the results obtained in our experiments, but this time ranked by their recall figures. The resulting rankings are now very different, providing us with new insights about the performance of the analyzed phonetic algorithms.

Overall, these results indicate that the *StatCan*, *Metaphone*, *Soundex* or *Roger Root* algorithms would be good choices for candidate generation in a microtext normalization system, depending on the level of compromise sought between precision and recall. This way, the *Metaphone* algorithm shines when precision is needed, as it is among the top–three algorithms in the F1 classification (see Tables 4 and 5) while it does not fall to the bottom of the recall rankings (Tables 6 and 7) as in the case of *MRA* or *Eudex*, the other two algorithms with the highest F1. On the other hand, *Soundex* and *Roger Root* stand as good choices if we want to maximize recall while not hurting precision in excess, as in

algorithm	#hits	avg #cands	P	R	F1
MRA	17,485	5.963	0.171	0.424	0.244
Eudex	16,150	3.682	0.174	0.392	0.241
Metaphone	22,448	16.640	0.136	0.545	0.218
Ref. Soundex	18,693	10.464	0.126	0.453	0.197
Beider–Morse	18,307	13.792	0.123	0.444	0.193
Rev. NYSIIS	17,844	16.149	0.091	0.433	0.150
F. Soundex	23,694	28.393	0.083	0.575	0.145
$Eudex_5$	18,498	42.203	0.085	0.449	0.143
StatCan	28,632	29.902	0.076	0.695	0.138
$Phonix_{most}$	19,039	23.382	0.081	0.462	0.137
Caverphone 2	21,580	43.374	0.075	0.524	0.132
NYSIIS	21,094	17.039	0.066	0.512	0.118
Caverphone 1	24,227	61.604	0.062	0.588	0.112
$Eudex_{10}$	19,988	150.981	0.054	0.485	0.097
Alpha SIS	26,064	145.874	0.048	0.632	0.089
D–M Soundex	24,470	64.662	0.045	0.594	0.084
$Phonix_{less}$	23,093	83.416	0.041	0.560	0.077
$Eudex_{15}$	20,892	290.397	0.038	0.507	0.071
Roger Root	28,928	98.075	0.028	0.702	0.054
Phonex	28,048	200.475	0.020	0.681	0.040
D. Metaphone	26,253	89.951	0.020	0.637	0.039
Soundex	29,557	94.946	0.018	0.717	0.035
Phonix Comm	26,937	134.754	0.018	0.654	0.035
$Phonix_{least}$	24,434	387.919	0.017	0.593	0.034
Onca	28,601	109.179	0.014	0.694	0.028
Lein	29,633	134.894	0.012	0.719	0.025
MRA_{custom}	38,768	20,767.367	6.873e-05	0.941	0.000

Table 5: Results for the `unimelb` dictionary, ranked by F1.

the case of the *Lein* algorithm, and after having dismissed MRA_{custom} for the reasons given above. Finally, *Statcan* strikes the best balance in precision and recall, as we can see in the good overall positions obtained in both rankings.

It is interesting to note that a phonetic algorithm can also be considered as a *locality–sensitive hashing algorithm*. These are algorithms that maximize the probability of a *collision* for similar items; this is, that the output for slightly different input elements is the same. In terms of phonetic algorithms, this is equivalent to maximizing the probability of two similar–sounding words having the same phonetic code. From this perspective, the average number of candidates retrieved by a phonetic algorithm is in fact directly proportional to the compression ratio of the locality–sensitive hashing it is performing.

During the error analysis, particular attention was paid to those instances from the corpora where none or just a few of the algorithms were able to provide the correct answer (in our context, a *hit*). In the case of the zero–hits list, i.e. instances for which no algorithm provided a correct answer, it contains examples such as `baddest–worst` or `5ayin–saying`, variations which do not correspond to phonetic phenomena and are thus outside the scope of this work. A few OOV words such as `thankyou` or `sheesh` also appear in this list. However, other interesting examples not directly supported by any phonetic algorithm

algorithm	#hits	avg #cands	P	R	F1
MRA_{custom}	3,695	15,270.327	0.000	0.929	0.000
Phonex	2,656	266.348	0.009	0.668	0.019
Roger Root	2,516	133.588	0.016	0.633	0.032
Lein	2,471	101.642	0.013	0.621	0.026
Soundex	2,469	80.570	0.015	0.621	0.030
Phonix Comm	2,453	194.972	0.011	0.617	0.023
Onca	2,396	109.237	0.010	0.602	0.020
D. Metaphone	2,395	84.389	0.014	0.602	0.028
Alpha SIS	2,359	241.792	0.022	0.593	0.042
$Phonix_{least}$	2,332	611.050	0.006	0.586	0.013
Caverphone 1	2,246	98.240	0.031	0.565	0.060
StatCan	2,225	21.415	0.070	0.559	0.124
$Eudex_{15}$	2,199	551.689	0.016	0.553	0.032
$Phonix_{less}$	2,191	141.575	0.020	0.551	0.040
F. Soundex	2,160	41.765	0.043	0.543	0.080
D–M Soundex	2,154	96.137	0.024	0.542	0.046
$Eudex_{10}$	2,076	310.535	0.025	0.522	0.048
Metaphone	2,071	26.232	0.078	0.521	0.137
Caverphone 2	2,039	69.022	0.039	0.513	0.073
$Eudex_5$	1,800	94.125	0.047	0.452	0.085
$Phonix_{most}$	1,681	36.503	0.044	0.422	0.080
Beider–Morse	1,636	25.514	0.080	0.411	0.134
Ref. Soundex	1,569	15.378	0.073	0.394	0.123
MRA	1,498	9.165	0.113	0.376	0.174
Rev. NYSIIS	1,416	27.404	0.047	0.356	0.083
NYSIIS	1,410	17.565	0.043	0.354	0.078
Eudex	1,376	5.641	0.117	0.346	0.175

Table 6: Results for the `utdallas` dictionary, this time ranked by recall.

can also be found, as is the case of the so–called *number homophones* (Thurlow, 2003) such as `2morrow`–`tomorrow` or `4got`–`forgot`. This is to be expected since this texting phenomenon, while also being a phonetic substitution, plays with the pronunciation of digits, and phonetic algorithms usually work only with letters. Furthermore, these examples would be easily supported in a microtext normalization system by preprocessing the input and spelling such numbers before applying the phonetic algorithm.

On the remaining lists, it is interesting to note the presence of instances where some phonetic algorithms gave a correct answer despite them not being designed to deal with that particular case. We can mention here those algorithms whose generated codes are shortened to a specific maximum length. Because of this shortening, such algorithms are able to cope with any misspellings occurring in those parts of the original word which are not finally translated into the phonetic code. For example, the MRA algorithm gives the correct answer `performance` for the input word `perfomence` as the second `r` is not encoded in `PRFMNC`.

Finally, taking into consideration the listing of non–standard orthographic forms from Thurlow (2003) and the lists of errors obtained in these experiments,

algorithm	#hits	avg #cands	P	R	F1
MRA_{custom}	38,768	20,767.367	6.873e-05	0.941	0.000
Lein	29,633	134.894	0.012	0.719	0.025
Soundex	29,557	94.946	0.018	0.717	0.035
Roger Root	28,928	98.075	0.028	0.702	0.054
StatCan	28,632	29.902	0.076	0.695	0.138
Onca	28,601	109.179	0.014	0.694	0.028
Phonex	28,048	200.475	0.020	0.681	0.040
Phonix Comm	26,937	134.754	0.018	0.654	0.035
D. Metaphone	26,253	89.951	0.020	0.637	0.039
Alpha SIS	26,064	145.874	0.048	0.632	0.089
D–M Soundex	24,470	64.662	0.045	0.594	0.084
$Phonix_{least}$	24,434	387.919	0.017	0.593	0.034
Caverphone 1	24,227	61.604	0.062	0.588	0.112
F. Soundex	23,694	28.393	0.083	0.575	0.145
$Phonix_{less}$	23,093	83.416	0.041	0.560	0.077
Metaphone	22,448	16.640	0.136	0.545	0.218
Caverphone 2	21,580	43.374	0.075	0.524	0.132
NYSIIS	21,094	17.039	0.066	0.512	0.118
$Eudex_{15}$	20,892	290.397	0.038	0.507	0.071
$Eudex_{10}$	19,988	150.981	0.054	0.485	0.097
$Phonix_{most}$	19,039	23.382	0.081	0.462	0.137
Ref. Soundex	18,693	10.464	0.126	0.453	0.197
$Eudex_5$	18,498	42.203	0.085	0.449	0.143
Beider–Morse	18,307	13.792	0.123	0.444	0.193
Rev. NYSIIS	17,844	16.149	0.091	0.433	0.150
MRA	17,485	5.963	0.171	0.424	0.244
Eudex	16,150	3.682	0.174	0.392	0.241

Table 7: Results for the `unimelb` dictionary, this time ranked by recall.

we can conclude the following:[8]

- *Shortenings* (e.g. `dec`–`december`, `epi`–`episode`) are difficult to account for as they usually remove whole chunks of characters from the original standard word, including consonants. The difficulty here is that consonants are the main building blocks for most phonetic codes and, consequently, important information for the algorithms has been stripped from the term to be normalized. This may be solved by adding an extra step in the normalization pipeline which would make use of other word similarity metrics such as the longest common subsequence, overlap coefficient or cosine distance (Okazaki and Tsujii, 2010).

- *Contractions* (e.g. `frm`–`from`, `lov`–`love`) can be dealt with as they generally include the most meaningful details of the standard word, which mainly consist of its consonants.

- *g–Clippings* (e.g. `losin`–`losing`, `frikin`–`freaking`) are a simple but problematic texting phenomenon for many of the algorithms studied. Most

[8]These lists are available at `http://www.grupocole.org/software/VCS/phon`

of them are able to incidentally handle it through their trimming of the output codes. In this way, in sufficiently long words the final `g` consonant is not taken into account for the encoding, hence bypassing the need for specific rules for managing it. On the other hand, it is worth noting that algorithms like *Beider–Morse* or *Phonex*, with a wide range of encoding rules, do handle this particular scenario effectively.

- Handling other types of *clippings* is possible for a wide range of algorithms if they perform some kind of clipping themselves during encoding (e.g. `luk`–`luck`, `metalic`–`metallic`) or when the clipping only affects vowels or non–pronounceable characters (e.g. `ther`–`their`, `oclock`–`o'clock`).

- *Acronyms* and *initialisms* (e.g. `omg`–`oh my god`, `lol`–`laughing out loud`) are not included in the evaluation datasets. In any case, they are considered to be outside the scope of phonetic phenomena and, consequently, are not consistently supported by any phonetic algorithm. Nevertheless, they may be normalized using specialized dictionaries (Doval et al., 2015; Han and Baldwin, 2011).

- In the case of *homophony* phenomena, letter homophones (e.g. `b`–`be`, `r`–`are`) are generally supported, whereas, as noted earlier, number homophones (e.g. `4got`–`forgot`, `in2`–`into`) are not.

- Other misspellings, typos, non–conventional spellings and accent stylization (e.g. `acount`–`account`, `basterds`–`bastards`, `huni`–`honey`, `dat`–`that`) can be handled by most algorithms as long as the sequence of consonants was preserved in the resulting non–standard word. In the case of examples as `eva`–`ever` or `ova`–`over`, they are only supported by the lowest precision algorithms or by those having specific rules for dealing with such phenomena.

5. Related work

As already mentioned in the introductory section of this work, phonetic algorithms have been traditionally used for personal name matching. In the literature, the design of a new phonetic algorithm tends to be coupled with a comparative study with the contemporary state of the art in order to highlight its strengths. However, in most cases it is not an exhaustive study, since only a small part of the existing phonetic algorithms are considered, and also because it focuses on the name–matching task, as in the case of the works of Hood (2004), Beider (2008), Mokotoff (2007), Holmes and McCabe (2002) and Parmar and Kumbharana (2014). Some exceptions in which a broader comparative study was performed are the studies made by Lynch and Arends (1977) and Lait and Randell (1996).

There are also purely comparative studies of name–matching algorithms where a wider range of techniques were considered, including phonetic algorithms and other types of similarity metrics. This is the case of the works

of Christen (2006), Snae (2007), Bilenko et al. (2003), Branting (2003) or Gálvez (2006). In their conclusions, these authors propose a series of recommendations to select the right approach or algorithm for a particular setup or domain, much in the same vein of the present work. However, even in these cases, they only compare a small subset of the phonetic algorithms available and, more importantly, they do so in the context of a different task.

Moving on from the name–matching task, the work of Pinto et al. (2012) enters the domain of microtexts. However, the authors only provide a comparative study between the original Soundex and their proposed improvements, which are not publicly available. Furthermore, their case is not that of microtext normalization either.

Finally, it is worth mentioning the recent work of Fuentes et al. (2016), which compares notably more phonetic algorithms than previous works although in a different scenario: word recognition in Spanish microtext mining. This may be the most similar work to the present contribution, although it focuses on precision and accuracy metrics, Spanish microtexts and, once again, it is not tailored to the microtext normalization task.

In general, these comparative studies mostly use precision and F1 metrics for their quantitative analysis, defining them accordingly for the task at hand. Some of them also give qualitative insights in order to exemplify the behaviour of the phonetic algorithms in each particular use case. On our end, we follow the common trend of using precision and F1, while also explicitly including the recall and other interesting measures such as the average number of normalization candidates retrieved by each algorithm. Likewise, we perform a qualitative study based on a classification of the texting phenomena. Overall, we present a wider comparison of phonetic algorithms than in previous work, focusing on the task of microtext normalization, while using tried and tested methods for performance measurements.

6. Conclusions and future work

In this work we have evaluated a wide range of English state–of–the–art phonetic algorithms within the context of generating normalization candidates in microtext normalization tasks. This work constitutes, to the best of our knowledge, the only wide–range comparative study of its kind. We perform both qualitative and quantitative analyses —adapting the usual performance metrics to our domain— in order to identify the most salient properties of these phonetic algorithms and their appropriateness for the task at hand. We expect that the results obtained will be of help to both developers and researchers of this field when building new intelligent systems for microtext information processing.

Seeking reproducibility and simplicity by using currently existing implementations and publicly available datasets, we have measured the performance of these algorithms, and their strengths and weaknesses were analysed to identify the best algorithms in terms of a compromise between precision and recall. In the end, we have found that the choice of phonetic algorithm depends heavily on

the capabilities of the subsequent candidate selection mechanism to be applied within the microtext normalization pipeline. The faster it can make the right selections among big enough input sets of candidates, the more we can sacrifice in terms of the precision of the phonetic algorithm in favour of coverage. This would be desirable since when obtaining a low number of normalization candidates, the system would run the risk of imposing an upper limit to its overall performance at this early stage.

Finally, as future lines of work, we plan to continue working on a more developed microtext normalization system and, in this way, improve the preliminary results obtained in our previous approach in this field (Doval et al., 2015). The results obtained in this study will greatly impact the design of the candidate selection method, which will be our main focus hereafter. The objective will be to obtain an accurate and efficient method that would allow us to take full advantage of wide–coverage phonetic algorithms in the candidate generation step. With respect to the limitations of the phonetic algorithms studied here when facing some texting phenomena, these can be tackled by other modules in the candidate generation step of the final normalization system. This may include, for instance, the use of spell checkers and non–standard–to–standard–text dictionaries (Doval et al., 2015; Han and Baldwin, 2011) or the integration of specialized word tokenizers (Doval et al., 2016). Extending this work to other languages such as Spanish (Alegria et al., 2015) or even code-switching scenarios (Vilares et al., 2016) is another possibility to be considered.

Acknowledgements

This research has been partially funded by the Spanish Ministry of Economy, Industry and Competitiveness (MINECO) through projects TIN2017-85160-C2-1-R, TIN2017-85160-C2-2-R, FFI2014-51978-C2-1-R and FFI2014-51978-C2-2-R, and by the Autonomous Government of Galicia through projects ED431D-2017/12, ED431B-2017/01 and ED431D R2016/046. Moreover, Yerai Doval is funded by the Spanish State Secretariat for Research, Development and Innovation (which belongs to MINECO) and by the European Social Fund (ESF) under a FPI fellowship (BES-2015-073768) associated to project FFI2014-51978-C2-1-R.

Alegria, I., Aranberri, N., Comas, P. R., Fresno, V., Gamallo, P., Padró, L., San Vicente, I., Turmo, J., Zubiaga, A., 2015. TweetNorm: a benchmark for lexical normalization of Spanish tweets. Language Resources and Evaluation 49 (4), 883–905.

Alegria, I., Aranberri, N., Fresno, V., Gamallo, P., Padró, L., San Vicente, I., Turmo, J., Zubiaga, A. (Eds.), 2013. TweetNorm 2013. Tweet Normalization Workshop 2013. Proceedings of the Tweet Normalization Workshop co-located with 29th Conference of the Spanish Society for Natural Language Processing (SEPLN 2013) Madrid, Spain, September 20th, 2013. Vol. 1086 of CEUR Workshop Proceedings. CEUR-WS.org. Corpus available at: `http://komunitatea.elhuyar.org/tweet-norm/` (accessed July 2018).

Apache Software Foundation, 2017. Apache Commons Codec. http://commons.apache.org/proper/commons-codec/ (accessed July 2018).

Atkinson, K., 2011. GNU Aspell. http://aspell.net (accessed July 2018).

Baldwin, T., Cook, P., Lui, M., MacKinlay, A., Wang, L., 2013. How noisy social media text, how diffrnt social media sources? In: Proc. of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013). Asian Federation of Natural Language Processing/ACL, pp. 356–364.

Baldwin, T., de Marneffe, M.-C., Han, B., Kim, Y.-B., Ritter, A., Xu, W., 2015. Shared Tasks of the 2015 Workshop on Noisy User-generated Text: Twitter Lexical Normalization and Named Entity Recognition. In: Proc. of the ACL 2015 Workshop on Noisy User-generated Text (W-NUT 2015). ACL, pp. 126–135. W-NUT 2015 website: http://noisy-text.github.io/2015/index.html (accessed July 2018).

Beaufort, R., Roekhaut, S., Cougnon, L.-A., Fairon, C., 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. In: Proc. of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010). ACL, pp. 770–779.

Beider, A., 2008. Beider-Morse phonetic matching: An alternative to Soundex with fewer false hits. Avotaynu: the International Review of Jewish Genealogy (Summer 2008).

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S., 2003. Adaptive name matching in information integration. IEEE Intelligent Systems 18 (5), 16–23.

Bisani, M., Ney, H., 2008. Joint–sequence models for grapheme–to–phoneme conversion. Speech communication 50 (5), 434–451.

Branting, L. K., 2003. A comparative evaluation of name–matching algorithms. In: Proc. of the 9th International Conference on Artificial Intelligence and Law (ICAIL'03). ACM, pp. 224–232.

Christen, P., 2006. A comparison of personal name matching: Techniques and practical issues. In: Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on. IEEE Press, pp. 290–294.

Doval, Y., Gómez-Rodríguez, C., Vilares, J., 2016. Spanish word segmentation through neural language models. Procesamiento del Lenguaje Natural 57, 75–82.

Doval, Y., Vilares, J., Gómez-Rodríguez, C. 2015. LYSGROUP: Adapting a Spanish microtext normalization system to English. In (Baldwin et al., 2015), pp. 99–105.

Duwairi, R. M., Marji, R., Sha'ban, N., Rushaidat, S., 2014. Sentiment Analysis in Arabic Tweets. In: 2014 5th International Conference on Information and Communication Systems (ICICS). IEEE Press.

Eisenstein, J., 2013. What to do about bad language on the Internet. In: Proc. of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013). ACL, pp. 359–369.

Eryiğit, G., Torunoğlu-Selamet, D., 2017. Social media text normalization for Turkish. Natural Language Engineering 23 (6), 835–875.

Foster, J., Cetinoglu, O., Wagner, J., Le Roux, J., Nivre, J., Hogan, D., van Genabith, J., 2011. From news to comment: Resources and benchmarks for parsing the language of Web 2.0. In: Proc. of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011). Asian Federation of Natural Language Processing/ACL, pp. 8–13.

Fuentes, A. A. G., Parra, I. P., Quevedo-Torrero, J. U., Perez, R. D., 2016. Comparative Analysis of Phonetic Algorithms Applied to Spanish. In: Computational Science and Computational Intelligence (CSCI), 2016 International Conference on. IEEE Press, pp. 1180–1185.

Gadd, T., 1988. 'Fisching fore werds': phonetic retrieval of written text in information systems. Program 22 (3), 222–237.

Gadd, T., 1990. Phonix: The algorithm. Program 24 (4), 363–366.

Gálvez, C., 2006. Identificación de nombres personales por medio de sistemas de codificación fonética. Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação 22, 105–116. Available at `http://www.redalyc.org/articulo.oa?id=14702209` (accessed July 2018).

Gerber, M. S., 2014. Predicting crime using Twitter and kernel density estimation. Decision Support Systems 61, 115–125.

Gill, L. E., Baldwin, J. A., 1987. Textbook of Medical Record Linkage. Oxford University Press, Ch. Methods and Technology of Record Linkage: Some Practical Considerations, pp. 39–54.

Gill, L., Goldacre, M., Simmons, H., Bettley, G., Griffith, M., 1993. Computerised linking of medical records: methodological guidelines. Journal of Epidemiology & Community Health 47 (4), 316–319.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., Smith, N. A., 2011. Part-of-speech Tagging for Twitter: Annotation, Features and Experiments. In: Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011) - Volume 2. ACL, pp. 42–47.

Han, B., Baldwin, T., 2011. Lexical normalisation of short text messages: makn sens a #twitter. In: Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011) - Volume 1. ACL, pp. 368–378.

Holmes, D., McCabe, M. C., 2002. Improving precision and recall for Soundex retrieval. In: Information Technology: Coding and Computing, 2002. Proceedings. International Conference on. IEEE Press, pp. 22–26.

Hood, D., 2002. Caverphone: Phonetic matching algorithm. Technical Paper CTP060902, University of Otago, New Zealand. Available at `http://caversham.otago.ac.nz/files/working/ctp060902.pdf` (accessed July 2018).

Hood, D., 2004. Caverphone revisited. Technical Paper CTP150804, University of Otago, New Zealand. Available at `http://caversham.otago.ac.nz/files/working/ctp150804.pdf` (accessed July 2018).

Jurafsky, D., Martin, J. H., 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2nd Edition. Pearson–Prentice Hall.

Karisani, P., Agichtein, E., 2018. Did you really just have a heart attack?: Towards robust detection of personal health mentions in social media. In: Proc. of the 2018 World Wide Web Conference (WWW'18). ACM, pp. 137–146.

Kaur, J., Singh, J., 2015. Toward Normalizing Romanized Gurumukhi Text from Social Media. Indian Journal of Science and Technology 8 (27).

Kell, J., 1988. NAMEHASH: Phonetic Name Indexing Routine. OS/2 REXX adaptation available at `http://www.markcrocker.com/rexxtipsntricks/rxtt28.2.0482.html` (accessed July 2018).

Kobus, C., Yvon, F., Damnati, G., 2008. Transcrire les SMS comme on reconnaît la parole. In: Actes de la 15e Conférence sur le Traitement Automatique des Langues (TALN 2008). pp. 128–138.

Lait, A. J., Randell, B., 1996. An assessment of name matching algorithms. Technical report, University of Newcastle upon Tyne, Department of Computing Science, United Kingdom. Available at `http://homepages.cs.ncl.ac.uk/brian.randell/Genealogy/NameMatching.pdf` (accessed July 2018).

Law, D., Gruss, R., Abrahams, A. S., 2017. Automated defect discovery for dishwasher appliances from online consumer reviews. Expert Systems with Applications 67, 84–94.

Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10 (8), 707–710.

Lynch, B. T., Arends, W. L., 1977. Selection of a surname coding procedure for the SRS record linkage system. Washington, DC: US Department of Agriculture, Sample Survey Research Branch, Research Division.

Madden, R., 2014. `stringmetric` package. `http://github.com/rockymadden/stringmetric` (accessed July 2018).

Mokotoff, G., 2007. Soundexing and genealogy. `http://www.avotaynu.com/soundex.html` (accessed July 2018).

Moore, G. B., 1977. Accessing individual records from personal data files using non–unique identifiers. Vol. 13. US Department of Commerce, National Bureau of Standards.

Odell, M., Russell, R. C., 1918. The Soundex coding system. US Patents 1261167.

Odell, M. K., 1956. The profit in records management. Systems 20 (20).

Okazaki, N., Tsujii, J., 2010. Simple and efficient algorithm for approximate dictionary matching. In: COLING'10: Proc. of the 23rd International Conference on Computational Linguistics. ACL, pp. 851–859.

Ølsgaard, M., 2016. `phonetic_search` package. `http://github.com/olsgaard/phonetic_search` (accessed July 2018).

Parmar, V. P., Kumbharana, C., 2014. Study Existing Various Phonetic Algorithms and Designing and Development of a working model for the New Developed Algorithm and Comparison by implementing it with Existing Algorithm(s). International Journal of Computer Applications 98 (19), 45–49.

Philips, L., 1990. Hanging on the Metaphone. Computer Language 7 (12), 39–43.

Philips, L., 2000. The Double Metaphone Search Algorithm. C/C++ Users Journal 18 (6), 38–43.

Pinto, D., Vilariño, D., Alemán, Y., Gómez, H., Loya, N., Jiménez-Salazar, H., 2012. The Soundex Phonetic Algorithm Revisited for SMS Text Representation. In: Text, Speech and Dialogue. Springer, pp. 47–55.

Plique, G., 2017. `talisman` package. `http://github.com/Yomguithereal/talisman` (accessed July 2018).

Ritter, A., Clark, S., Mausam, Etzioni, O., 2011. Named entity recognition in tweets: An experimental study. In: EMNLP'11: Proc. of the Conference on Empirical Methods in Natural Language Processing. ACL, pp. 1524–1534.

Rudra, K., Banerjee, S., Ganguly, N., Goyal, P., Imran, M., Mitra, P., 2016. Summarizing situational tweets in crisis scenario. In: Proc. of the 27th ACM Conference on Hypertext and Social Media (HT'16). ACM, pp. 137–147.

Saralegi, X., San Vicente, I., 2013. Elhuyar at Tweet-Norm 2013. In (Alegria et al., 2013).

Schulz, S., Pauw, G. D., Clercq, O. D., Desmet, B., Hoste, V., Daelemans, W., Macken, L., 2016. Multimodular Text Normalization of Dutch User-Generated Content. ACM Transactions on Intelligent Systems and Technology (TIST) 7 (4), 61:1–61:22.

Snae, C., 2007. A comparison and analysis of name matching algorithms. International Journal of Applied Science. Engineering and Technology 4 (1), 252–257.

Taft, R. L., 1970. Name search techniques. Special Report no. 1, Bureau of Systems Development, New York State Identification and Intelligence System, Albany, NY.

Thurlow, C., 2003. Generation Txt? The sociolinguistics of young people's text–messaging. Discourse Analysis Online 1 (1). Available at `http://extra.shu.ac.uk/daol/articles/v1/n1/a3/thurlow2002003-paper.html` (accessed July 2018).

Ticki, 2016. Eudex: A blazingly fast phonetic reduction/hashing algorithm. `http://github.com/ticki/eudex` (accessed July 2018).

Vilares, D., Gómez-Rodríguez, C., Alonso, M. A., 2017. Universal, unsupervised (rule-based), uncovered sentiment analysis. Knowledge-Based Systems 118, 45–55.

Vilares, D., Alonso, M. A., Gómez-Rodríguez, C., 2016. EN-ES-CS: An English-Spanish Code-Switching Twitter Corpus for Multilingual Sentiment Analysis. In Proc. of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA).

Vilares, D., Thelwall, M., Alonso, M. A., 2015. The Megaphone of the People? Spanish SentiStrength for Real-time Analysis of Political Tweets. Journal of Information Science 41 (6), 799–813.

Wang, P., Ng, H. T., 2013. A Beam-Search Decoder for Normalization of Social Media Text with Application to Machine Translation. In: Proc. of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013). ACL, pp. 471–481.

Xue, Z., Yin, D., Davison, B. D., 2011. Normalizing microtext. In: Analyzing Microtext, Papers from the 2011 AAAI Workshop, San Francisco, California, USA, August 8, 2011. Technical Report WS-11-05. AAAI Press.