A Syllabification Algorithm for Spanish

Heriberto Cuayáhuitl

Universidad Autónoma de Tlaxcala,
Department of Engineering and Technology,
Intelligent Systems Research Group,
Apartado Postal #140, Apizaco, Tlaxcala, Mexico, 90300.
hcuayahu@ingenieria.uatx.mx

Abstract. This paper presents an algorithm for dividing Spanish words into syllables. This algorithm is based on grammatical rules which were translated into a simple algorithm, easy to implement and with low computational cost. Experimental results in an evaluation text corpus show an overall error rate of 1.6%. Most of the error is attributed to words with diphthongs and to confusion in the use of prefixes where grammatical rules are not always absolute. Syllabification is an essential component of many speech and language processing systems, and this algorithm might be very useful to researchers working with the Spanish language.

1 Introduction

Currently, the development of speech synthesizers and speech recognizers, frequently requires working with subword units such as syllables [1-3]. For instance, robust speech recognition often makes use of word spotters based on syllables for detecting Out-of-Vocabulary (OOV) speech [2] and for modeling unknown words in spontaneous speech [3]. Today, every new system being developed requires the implementation of a new algorithm for dividing words into syllables, due to the fact that a formal algorithm shared among the linguistic community does not exist, at least for Spanish. In the linguistic literature, we can find grammatical rules or attempts to explain the division of words into syllables step-by-step, but nothing beyond that. In the past, syllabification algorithms have been proposed for different languages, including English and German, among others [4], implemented as a weighted finite state transducer, but this is not the case for Spanish, where few research efforts have been documented. Thus, the purpose of this work is to formulate an algorithm for dividing Spanish words into syllables, and to share this algorithm so that other researchers in the area of speech and language processing will not have to duplicate the work.

In this research, is proposed an algorithm to divide Spanish words into syllables. Our experiments are based on a text corpus containing representative words for each grammatical rule. Results are given in terms of a simple division of correctly syllabified words by the total number of words. In the remainder of this paper we first provide an overview of Spanish syllabification in section 2. In section 3 we describe the syllabification algorithm itself. In section 4 we present experimental results. Finally, we provide some conclusions and future directions.

A. Gelbukh (Ed.): CICLing 2004, LNCS 2945, pp. 412–415, 2004.

2 Syllabification in Spanish

Spanish letters are either vowels (*a,e,i,o,u*) or consonants (*b,c,d,f,g,h,j,k,l,ll,m,ñ, n,p,q,r,rr,s,t,v,w,x,y,z*), and vowels are either weak (*i,u*) or strong (*a,e,o*). Letters *ch*, *ll*, and *rr* are considered as single consonants. In order to illustrate the syllabification process [5], the following steps were considered for creating the algorithm:

1. Scan the word from left to right
2. If the word begins with a prefix, divide between the word and the prefix
3. Ignore one or two consonants if they begin a word
4. Skip over vowels
5. When you come to a consonant, see how many consonants are between vowels
   a) If there is only one, divide to the left of it;
   b) If there are two, divide to the left of the second one, but if the second one is *l* or *r*, divide to the left of the first one;
   c) If there are three, divide to the left of the third one, but if the third one is *l* or *r*, divide to the left of the second one;
   d) If there are four, the fourth one will always be *l* or *r*, so divide before the third consonant.
6. If the consonant ends the word, ignore it
7. Scan the word a second time to see if two or more vowels are together
   a) If two vowels together are both weak, ignore them;
   b) If one of the vowels is weak, ignore it, but if the *u* or *i* has an accent mark, divide between the two vowels;
   c) If only one of the vowels is weak, and there is an accent mark which is not on the *u* or *i*, ignore them;
   d) If both vowels are strong, divide between the vowels;
   e) If there are three vowels together, ignore them if two of them are weak even if there is an accent mark; if two of the three vowels are strong, separate the two strong vowels if they are side by side.

3 The Syllabification Algorithm

The syllabification algorithm (figure 1) basically follows the steps provided above and is written with a neutral notation so that it can be implemented in virtually any programming language. For implementing a syllabifier, following the algorithm should be easier and faster than following the textual description provided in section 2, because coding is straightforward. The complexity of this algorithm is *O(n+m)*, where *n* is the number of entries in the prefix list, and *m* is the number of characters in the string. Prefixes played an important role in the algorithm due to the fact that we had to keep these subword units together. Here is the list of prefixes used in our experiments: *circun, cuadri, cuadru, cuatri, quinqu, archi, arqui, citer, cuasi, infra, inter, intra, multi, radio, retro, satis, sobre, super, supra, trans, ulter, ultra, yuxta, ante, anti, cata, deci, ecto, endo, hemi, hipo, meta, omni, pali, para, peri, post, radi, tras, vice, cons, abs, ana, apo, arz, bis, biz, cis, com, con, des, dia, dis, dis, epi, exo, met, pen, pos, pre, pro, pro, tri, uni, viz, ins, nos.*

algorithm **Syllabifier**(String S) **return** (**N+T**)

$P = \{x \mid x \text{ is a } prefix\}$, $V = \{x \mid x \text{ is a vowel}\}$, $C = \{x \mid x \text{ is a consonant}\}$
$V_s = \{x \mid x \text{ is a strong vowel}\}$, $V_w = \{x \mid x \text{ is a weak vowel}\}$
$V_{wa} = \{x \mid x \text{ is a weak accented vowel}\}$, $N = T = ""$, $i = 0$

for $x = 0, \ldots, \mid P \mid$-1 **do**
  if $(P_x \in S)$ **then**
    $N \leftarrow P_x + " - "$
    $i \leftarrow \mid P_x \mid$
    break
  end if
end for

for $i = i, \ldots, \mid S \mid$-1 **do**
  if $(((S_{i-1} \in V_s) \vee (S_{i-1} \in V_{wa})) \wedge (S_i \in V_s)) \vee ((S_{i-1} \in V) \wedge (S_i \in V_{wa}))$ **then**
    $N \leftarrow N + " - "$
    $T \leftarrow S_i$
    continue
  end if

  if $((S_i \in C) \wedge (S_{i+1} \in V)) \vee ((S_{i+1} \in V_a) \wedge (T \neq \emptyset))$ **then**
    if $(S_i \in \{l, r\}) \wedge (S_{i-1} \in C)$ **then**
      if $(i > 1)$ **then**
        $T \leftarrow T_k, \forall\ 0 \leq k \leq (\mid T \mid -1)$
        $N \leftarrow N + T + " - "$
        $T \leftarrow S_{i-1} + S_i$
      else
        $T \leftarrow T + S_i$
      end if
    else
      $N \leftarrow N + T + " - "$
      $T \leftarrow S_i$
    end if
  else
    $T \leftarrow T + S_i$
  end if
end for

Fig. 1. The Syllabification Algorithm. Sample output given the same text without divisions (words in bold font were incorrectly divided): *Cier-to hom-bre, que ha-bí-a com-pra-do u-na va-ca mag-ní-fi-ca, so-ñó la mis-ma no-che que cre-cí-an a-las sobre la es-pal-da del a-ni-mal, y que és-te se mar-cha-ba vo-lan-do. Con-si-de-ran-do es-to un pre-sa-gio de in-for-tu-nio in-mi-nen-te, lle-vó la va-ca al mer-ca-do nue-va-men-te, y la ven-dió con gran pér-di-da. En-vol-vien-do en un pa-ño la pla-ta que re-ci-bió, la e-chó sobre su es-pal-da, y a mi-tad del ca-mi-no a su ca-sa, vio a un hal-cón co-mien-do par-te de u-na lie-bre. A-cer-cán-do-se al a-ve, des-cu-brió que e-ra bas-tan-te man-sa, de ma-ne-ra que le a-tó u-na pa-ta a u-na de las es-qui-nas con pa-ño en que es-ta-ba su di-ne-ro. El hal-cón* ***a-le-te-a-ba*** *mu-cho, tra-tan-do de es-ca-par, y tras un ra-to, al a-flo-jar-se* ***mo-men-tá-ne-a-men-te*** *la ma-no del hom-bre, vo-ló con to-do y el tra-po y el di-ne-ro. "Fue el des-ti-no", di-jo el hom-bre ca-da vez que con-tó la his-to-ria; ig-no-ran-te de que, pri-me-ro, no de-be te-ner-se fe en los sue-ños; y, se-gun-do, de que la gen-te no de-be re-co-ger co-sas que ve al la-do del ca-mi-no. Los cua-drú-pe-dos ge-ne-ral-men-te no vue-lan.*

4 Experiments and Results

For detecting errors in the preliminary version we used representative words for each rule provided by [5]. The evaluation text corpus consisted of 316 words ranging from one to six syllables, extracted from [6-8]. Part of this evaluation corpus is shown in figure 1. The evaluation was performed with a simple division between the number of correctly syllabified words and the total number of words. Our results show a 98.4% of accuracy where most of the error can be attributed to words with diphthongs and to confusion in the application of prefixes, so we could see that grammatical rules are not absolute. For instance, one of the grammatical rules says that "prefixes should remain intact", but although there is a prefix *extra*, the word *extraer* should be syllabified as *ex-tra-er*.

5 Conclusions and Future Work

In this paper is presented an algorithm for dividing Spanish words into syllables. The algorithm is based on grammatical rules proposed by [5] and does not require high computational cost. Because of its simplicity, the implementation of this algorithm into several programming languages should be feasible with minimal effort. we recommend this simple, easy to implement, and accurate algorithm to the community interested in the area of speech and language processing for Spanish. An immediate future work consist in resolving current problems of this algorithm in order to provide an accurate algorithm to the linguistic community. Perhaps, the Porter Stemming Algorithm [9], used to find root words, may help to detect if a word has a prefix. Also we plan the incorporation of this algorithm to the generation of a robust speech recognizer for dealing with OOV speech.

References

[1] Black, A. and Lenzo, K.: Optimal Data Selection for Unit Selection Synthesis. In proceedings of the 4th Speech Synthesis Workshop, Scotland (2001)
[2] Cuayáhuitl, H. and Serridge, B.: Out-Of-Vocabulary Word Modeling and Rejection for Spanish Keyword Spotting Systems. In proceedings of the MICAI, Springer-Verlag, LNAI 2313, Merida, Mexico (2002) 158-167
[3] Kemp, T. and Jusek, A.: Modeling Unknown Words in Spontaneous Speech. In Proceedings of ICASSP, Atlanta, GA (1996) 530-533
[4] Kiraz, G. A. and Mobius, B.: Multilingual syllabification using weighted finite-state transducers. In Proceedings of the Third ESCA Workshop on Speech Synthesis, Jenolan Caves, Australia (1998)
[5] Glenn Humphries: Syllabification: The Division of Words into Syllables. http://glenn.humphries.com/Notebook/toc.htm
[6] Uzcanga, A. M.: La Ortografía es fácil. EDAMEX, ISBN 968-409-914-2 (2000)
[7] Mungia, I., Mungía, M. E., and Rocha, G.: Gramática de Lengua Española - Reglas y Ejercicios. LAROUSSE, ISBN 970-22-0058-X (2000)
[8] Maqueo, A. M.: Ortografía. LIMUSA, ISBN 968-18-1547-5 (2002)
[9] Jurafsky, D. and Martin, J.H.: Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, Upper Saddle River, 2000.