UNA PROPUESTA PARA EL ANÁLISIS MORFOLÓGICO DE VERBOS DEL ESPAÑOL

A PROPOSAL FOR MORPHOLOGICAL ANALYSIS OF VERBS IN THE SPANISH LANGUAGE

CARLOS MARIO ZAPATA
Grupo de Investigación en Lenguajes Computacionales, Universidad Nacional de Colombia, cmzapata@unal.edu.co

JHON EDISON MESA
Grupo de Investigación en Lenguajes Computacionales, Escuela de Sistemas, Universidad Nacional de Colombia

Recibido para revisar abril 27 de 2007, aceptado diciembre 14 de 2007, versión final enero 16 de 2008

RESUMEN: El análisis morfológico de verbos del español no es una tarea fácil, debido a sus peculiares características. Una de esas características es la gran cantidad de conjugaciones que puede poseer un verbo, lo cual dificulta la generación automática de dichas conjugaciones. Aunque, en la actualidad, existen propuestas de analizadores de verbos que realizan la conjugación y lematización de estos, aún presentan problemas. En este artículo se propone un analizador morfológico que utiliza plantillas para generalizar las conjugaciones de los verbos. Con estas plantillas se pueden realizar los procesos de conjugación y lematización, sin tener que almacenar todas las posibles conjugaciones de un verbo de manera independiente. La implementación del analizador morfológico de verbos se hizo en el lenguaje de programación Python para aprovechar las capacidades que éste ofrece en la manipulación de cadenas de texto.

PALABRAS CLAVE: Conjugación, Lematización, Verbos Modelo, Análisis Morfológico, Plantilla, Generación.

ABSTRACT: The morphological analysis of verbs in the Spanish language is a difficult task due to the fact that they present special features. One of these features is related to the big amount of conjugation forms of a verb, and this fact makes difficult the automatic generation of these conjugations. Currently, there are proposals of verb analyzers, which make lemmatization and conjugation of verbs, but they still exhibit some problems. In this article, we propose a template-based morphological analyzer with the purpose of generalizing verb conjugations. The templates aim to make the conjugation and lemmatization processes without the need of independently saving all possible conjugation forms of a verb. The morphological analyzer was implemented in the programming language Python, to take advantage of the capabilities of this language for string manipulation.

KEY WORDS: Conjugation, Lemmatization, Model Verbs, Morphological Analysis, Templates, Generation.

1. INTRODUCCIÓN

El análisis morfológico es el estudio de la estructura de las palabras de un idioma identificando en ellas sus respectivos morfemas [1]. Concretamente, en los verbos del español el análisis morfológico consiste en identificar en un verbo la raíz y el prefijo flexivo que cambian la función gramatical de la palabra raíz [2]. El objetivo de un análisis morfológico de verbos, cuando se realiza de manera automática, es encontrar la flexión de un verbo en infinitivo en cualquier tiempo y conseguir lematizarlo a su forma de infinitivo [1].

Tomando como base el español, el análisis morfológico de verbos no es una tarea fútil debido a sus peculiares características, tales como: sus variados accidentes verbales (gracias a la existencia de los múltiples tiempos, personas y números implicados en la conjugación), los diferentes paradigmas verbales, la división de los verbos en regulares e irregulares, los verbos defectivos y la presencia del participio irregular [3].

En la actualidad, existen analizadores morfológicos que permiten conjugar y lematizar verbos en español, pero todavía persisten problemas tales como los siguientes:

- Conjugación incompleta de ciertos verbos, con varias alternativas de conjugación en un mismo tiempo, persona y número [4].
- Necesidad de grandes bases de datos para su correcto funcionamiento, para almacenar los verbos en infinitivo con sus correspondientes conjugaciones, lo cual genera ineficiencias en el análisis morfológico.
- Fallas en la conjugación y lematización de ciertos verbos, tales como la presentación de resultados erróneos o la incapacidad de encontrar el correspondiente verbo [5].
- Soluciones teóricas que no se llegan a implementar, *e. g.* Zamorano [7].

En este artículo se presenta una propuesta de solución que permite obtener la conjugación y lematización de un verbo del español usando un lexicón de verbos modelo, que agrupa las regularidades e irregularidades de un gran conjunto de verbos. La solución propuesta conduce el análisis morfológico a la concatenación o separación de una raíz y una terminación según un tiempo, persona o número deseado. Además, permite aliviar las grandes y poco eficientes bases de datos que requieren algunos conjugadores, tratando de solucionar algunos problemas de la literatura especializada en el tema.

El artículo tiene la siguiente estructura: en la Sección 2 se discute sobre el funcionamiento de algunos analizadores morfológicos de verbos destacando sus ventajas y desventajas; en la Sección 3 se describe un marco teórico de la propuesta de solución; la propuesta de solución como tal se presenta en la Sección 4; finalmente, las conclusiones y trabajo futuro se discuten en la Sección 5.

2. REVISIÓN DE LA LITERATURA EN ANALIZADORES MORFOLÓGICOS DE VERBOS

Algunos analizadores morfológicos de verbos se describen a continuación:

2.1 Flaver

El flexionador y lematizador automático de formas verbales [5, 6] permite lematizar diferentes verbos, identificando en este proceso su infinitivo, categoría gramatical y flexión, y generando una forma verbal a partir de su infinitivo. Este sistema posee una estructura de datos conformada por verbos modelo, terminaciones y reglas, que le permiten llevar a cabo sus procesos de forma bidireccional sobre la misma estructura.

La ventaja de este sistema es la estructura de datos, que guarda la información mínima necesaria para generar las conjugaciones, como los infinitivos, sus verbos modelos asociados y las reglas de cambio de raíz. Por ello, esta estructura permite disminuir los tiempos de lematización que se requerirían con una base de datos que guarda a cada verbo sus flexiones según tiempo, persona y número.

La desventaja de esta solución es el manejo de reglas para los cambios de raíz, el cual requiere mucho tiempo y esfuerzo para construir los algoritmos necesarios que aplican dichas reglas sobre un verbo [4]. Además, presenta algunas fallas en la conjugación de verbos defectivos como el verbo “embaír”, cuyo resultado se aprecia en la Figura 1 y que no debería presentar la tercera persona del singular y del plural en el pretérito indefinido de indicativo. La flexión correcta de “embaír”, se puede consultar en la página oficial de la Real Academia Española RAE [8].

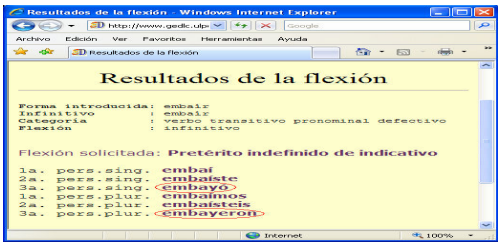

Figura 1. Conjugación de “embaír” en FLAVER
Figure 1. “*embaír”* conjugation by FLAVER

2.2 Propuesta de estructura léxica para el análisis morfológico de verbos

Zamorano [7] define la estructura de una gramática, que permite obtener la conjugación automática de cualquier verbo, contemplando las irregularidades

que se puedan presentar en la raíz y en la terminación, en un tiempo, persona y número específico.

La ventaja de esta propuesta, al igual que el sistema de Santana *et al.* [5], es la reducción del léxico necesario para lograr la conjugación de un verbo, además de que no se requiere especificar reglas de cambio de raíz.

La desventaja de la propuesta de Zamorano [7] es que fue diseñada para un entorno con propósitos de enseñanza académica. Además, la estructura léxica definida por el autor no permite precisar de forma intuitiva, el proceso de lematización y se limita a un estudio teórico de la conjugación de los verbos que no se llevó a una implementación.

2.3 AGME (Análisis y Generación morfológica para el Español)

AGME [9] usa un diccionario de raíces que se asocian con el verbo al que pertenecen. Este sistema agrupa los verbos en tres conjuntos: regulares, semirregulares e irregulares.
Una ventaja de este sistema, es el uso de raíces, lo cual reduce el problema de la conjugación a buscar la raíz apropiada y concatenarla con la terminación respectiva. Otra ventaja que posee es la obtención, con una misma estructura, de la conjugación y la lematización de verbos.
La desventaja de este sistema, es que presenta algunas fallas en la lematización de ciertos verbos, que tienen la particularidad de poseer hasta tres alternativas de conjugación en mismo tiempo, persona y número. Por ejemplo, el verbo "yacer", en el tiempo Presente Indicativo, primera persona del singular, se puede conjugar como "yazco", o "yazgo", o "yago". AGME reconoce la forma "yazgo", pero las otras dos no (véase la Figura 2). La flexión correcta de "yacer" se puede encontrar en la página oficial de la RAE [8].

3. MARCO TEÓRICO

3.1 Teoría de verbos

Para el idioma español, el análisis morfológico de verbos no es una tarea simple debido a sus peculiares características: "cada tiempo contiene tres formas distintas para cada persona y, a su vez, cada persona puede aparecer en singular o plural" [7]. Además, el español posee en total 62 formas verbales simples asociadas a cada verbo, además de las 45 formas de conjugación compuestas para un total general de 107 formas verbales asociadas a cada verbo. [10].

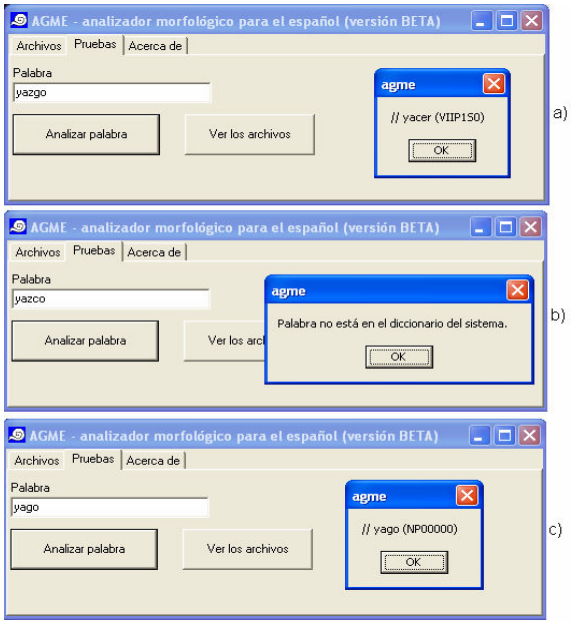

Figura 2. Conjugación de "yacer" en AGME
Figure 2. *"yacer"* conjugation by AGME

Aunque las anteriores características pueden ser computacionalmente tratables, otras características le dan a la morfología verbal la connotación compleja para el análisis automático, tales como:

- Existencia de tres paradigmas de conjugación (verbos terminados en -ar, -er, -ir) [10].
- Existencia de verbos irregulares los cuales, cuando se conjugan, sufren cambios en la raíz [11].
- Existencia de irregularidades en las terminaciones de algunos tiempos [12].
- Presencia de vacíos en algunos paradigmas de conjugación de ciertos verbos, ya que dicha conjugación no se usa o no existe [3].
- Presencia de verbos que poseen doble o triple alternativa de conjugación en un mismo tiempo persona y número [12].
- Existencia de algunos verbos que poseen doble o triple participio pasado [3].

La conjugación verbal se suele agrupar bajo el contexto de los verbos modelo, según las

irregularidades que presenten. Para los verbos regulares, se agrupan según la terminación del infinitivo. Para los verbos irregulares, los verbos modelo se crean según las irregularidades que presenten [11]. Aunque la RAE [8] reconoce 62 verbos modelo, el Instituto de Verbología Hispánica [11], identificó 101 modelos verbales que agrupan casi todos los verbos existentes.

3.2 Lenguaje de programación Python

Python es un lenguaje de programación, con una serie de características que lo hacen ideal para aplicaciones que requieran un alto manejo de cadenas de caracteres, tales como:

- Expresividad: un programa Python suele ser bastante más corto que su equivalente en lenguajes como C [13].
- Legibilidad: La sintaxis de Python permite la escritura de programas cuya lectura resulta más fácil que en otros lenguajes de programación [13].
- Versatilidad: Python se puede usar como imperativo procedimental o como lenguaje orientado a objetos [13].
- Python posee dos enfoques de creación aplicaciones. El primer enfoque se centra en la creación de aplicaciones de escritorio y el segundo enfoque en la creación de aplicaciones web.

Gracias a las anteriores ventajas, Python permite implementar aplicaciones con alto grado de manipulación de *strings*, como la que se desarrolla en esta propuesta.

4. PROPUESTA DE SOLUCIÓN: ANALIZADOR MORFOLÓGICO DE VERBOS EN ESPAÑOL

En este artículo se propone la definición de una estructura léxica guiada por verbos modelo, que permite realizar los dos procesos principales: conjugación y lematización. Esta solución se enfoca en el uso de un diccionario de raíces para llevar a cabo los dos procesos, por lo que no se requiere entrar a definir reglas de cambio de raíz.

4.1 Estructura léxica

Para cada verbo modelo, se llena la plantilla que se muestra a continuación, cuya sintaxis es similar a la expuesta por Zamorano [7].

```
( LEXICAL-VERB-ITEM
: NAME ""
: TYPE ""
: ROOT ""
: CONJUGATION-MODEL ""
: CONJUGATION-TERM "")
```

La etiqueta NAME guarda el nombre del verbo modelo (*e. g.* “amar”, “haber”, etc.). La etiqueta TYPE indica el grupo al que pertenece el verbo modelo (regulares o isoptongos, auxiliares, semirregulares, irregulares y tíldicos). La etiqueta ROOT indica la raíz o raíces que posee el verbo modelo. Si posee mas de una raíz, éstas se separan con el símbolo “|” (*e. g.* hab|h|hay|hub). La etiqueta CONJUGATION-MODEL indica el modelo de conjugación que sigue el verbo modelo. Un vector de 60 posiciones representa el modelo; cada posición se asocia con un tiempo, persona y número en particular y el código que se almacena en cada posición determina la raíz a usar en la conjugación. El código de la raíz se asigna según la posición que ocupe en ROOT. En la Tabla 1 aparecen los tiempos que se reconocen. En algunos verbos modelo, es posible que, en un tiempo, persona y número en particular, se usen varias raíces. Este caso se representa en el modelo de conjugación con las raíces que se usan encerradas entre “[ ]”. Por ejemplo, para el verbo modelo “haber” en el tiempo Presente Indicativo, primera persona del plural se deben usar las raíces uno y dos. Esto es correcto ya que las formas “habemos” y “hemos” son válidas [11].

Adicional a lo anterior, algunos verbos modelo no usan una raíz en la conjugación de algunos tiempos, por lo que el código de la raíz es cero (0). Como en el Imperativo, la primera persona del singular, y en las formas impersonales, la primera, segunda y tercera del plural no existen, se debe colocar la raíz cero (0) y asociarla con la terminación *dummy*, que se describe luego.

La etiqueta CONJUGATION-TERM indica el código del grupo de terminaciones que usa el verbo modelo para conjugar en un tiempo particular. Cada código apunta a otro creado en la plantilla “Asociación terminación”, que se describe luego. En total son 10 códigos para 10 tiempos y cada código separado por el símbolo “::”.

Tabla 1. Convención para TIME, PERSON y NUMBER
Table 1. Convention for TIME, PERSON, and NUMBER tags

ETIQUETA	CÓDIGO
TIME	1: Presente indicativo
	2: Presente Subjuntivo
	3: Imperativo
	4: Pretérito Indefinido
	5: Pretérito Imperfecto de Subjuntivo
	6: Futuro Imperfecto de Subjuntivo
	7: Futuro Imperfecto de indicativo
	8: Condicional Simple
	9: Pretérito Imperfecto de Indicativo
	10: Formas Impersonales
PERSON	1: Primera Persona
	2: Segunda Persona
	3: Tercera Persona
NUMBER	1: Singular
	2: Plural

Por ejemplo, para el verbo "haber", la plantilla se vería así:

```
( LEXICAL-VERB-ITEM
: NAME "haber"
: TYPE "auxiliar"
: ROOT "hab|h|hay|hub"
: CONJUGATION-MODEL
"222[12]123333330233131111114444444444411111111111
1111111111000"
: CONJUGATION-TERM
"A70::A12::A13::A39::A15::A16::A52::A53::A19::A20")
```

Para todas las terminaciones (regulares e irregulares) que existen en la conjugación se llena la siguiente plantilla:

```
( TERMINATION-ITEM
: CODE ""
: NAME ""
: TIME ""
: PERSON ""
: NUMBER "")
```

La etiqueta CODE indica el código de la terminación. El código empieza por la letra "T" seguido por un número consecutivo. La etiqueta NAME indica la terminación como tal (*e. g.* -o, -oy, etc). Es posible que en un tiempo, persona y número en particular se usen varias terminaciones según un verbo modelo. Estas terminaciones se colocan en NAME separadas por el símbolo "|". Las etiquetas TIME, PERSON y NUMBER, indican el tiempo, persona y número en donde se usa la terminación. Por convención, en la Tabla 1 se encuentra una codificación para TIME, PERSON y NUMBER.

Por ejemplo, para "-oy", la plantilla se vería así:

```
( TERMINATION-ITEM
: CODE "T130"
: NAME "oy"
: TIME "1"
: PERSON "1"
: NUMBER "1")
```

Las terminaciones ingresadas con la plantilla anterior se agrupan después en la plantilla "Asociación Terminación" según el tiempo al que pertenezcan. La plantilla "Asociación Terminación" es la siguiente:

```
( TERMINATION-ASSOCIATION
: CODE ""
: TIME ""
: CODES-ASSOCIATION "")
```

Adicional a las terminaciones listadas, se debe ingresar una terminación *dummy*, que indica el no uso de terminación en un tiempo, persona y número particular. Su plantilla es la siguiente:

```
 ( TERMINATION-ITEM
: CODE "T26"
: NAME "*"
: TIME "*"
: PERSON "*"
: NUMBER "*")
```

Un ejemplo de llenado de la plantilla "Asociación Terminación" es el siguiente:

```
( TERMINATION-ASSOCIATION
: CODE "A1"
: TIME "1"
: CODES-ASSOCIATION "T1::T2::T3::T4::T5::T6")
```

La etiqueta CODE indica el código de la terminación. El código empieza por la letra "A" seguido por un número consecutivo. La etiqueta TIME indica el tiempo donde opera la asociación. Los valores de esta etiqueta se encuentran en la Tabla 1. La etiqueta CODES-ASSOCIATION

indica el conjunto de terminaciones que se relacionan con la asociación (plantilla). Es un vector de seis posiciones en donde cada posición almacena el código de la terminación, separada por el símbolo "::". Cabe resaltar que el valor que guarden las terminaciones en la etiqueta TIME debe coincidir con el valor guardado en la etiqueta TIME de la asociación.

El diccionario de raíces es una tabla en donde cada verbo del español se relaciona con su verbo modelo y se almacenan las raíces que tiene. Además, se almacena el tipo de verbo defectivo y el tipo de irregularidad en el participio que posee. Un ejemplo del diccionario de raíces se encuentra en la Tabla 2. En la columna "raíces" se colocan las raíces de la plantilla "Verbo Modelo". La columna "Tipo de verbo defec" almacena el tipo de verbo defectivo según el criterio definido por Santana *et al.* [5]. Si el verbo no es defectivo, se coloca "d0" o se deja el espacio en blanco (*i.e.* "[]"). La columna "Tipo de irregularidad participio" guarda las irregularidades que tenga el verbo en el participio. Si el verbo posee dicha irregularidad, se almacenan todos sus participios irregulares. Si el verbo posee más de un participio, estos se separan con el símbolo "|". Si el verbo no posee irregularidad en el participio, se deja el espacio en blanco (*i.e.* "[]"). Es pertinente aclarar que, si se especifica esta irregularidad, cuando se busque la conjugación del Participio se elijen las ingresadas en el diccionario y no las generadas con el verbo modelo.

Tabla 2. Diccionario de raíces
Table 2. Root dictionary

verbo	verbo modelo	raíces	tipo verbo defec	tipo de irregularidad participio
deber	deber	[deb]	[]	[]
huir	construir	[hu|huy]	[d0]	[]
nevar	acertar	[nev|niev]	[d2]	[]
abrir	vivir	[abr]	[d0]	[abierto]

Para los verbos que no es posible asociar a algún verbo modelo, se llena la siguiente plantilla:

```
( LEXICAL-VERB-ITEM
: NAME ""
: CONJUGATION-MODEL "")
```

La etiqueta NAME contiene el infinitivo al que se le llena la plantilla. La etiqueta CONJUGATION-MODEL contiene todas las conjugaciones del verbo en los 10 tiempos. Cada conjugación se separa con el símbolo "|". Si el verbo posee varias alternativas de conjugación en un tiempo, persona y número en particular, éstas se separan con el símbolo "&"; y si el verbo no posee conjugación se representa con el símbolo "*".
Por ejemplo, para el verbo atípico "ir", la plantilla es la siguiente:

```
( LEXICAL-VERB-ITEM
: NAME "ir"
:CONJUGATION-MODEL
"voy|vas|va|vamos|vais|van|vaya|vayas|vaya|vayamos|vayáis|va
yan|*|ve|vaya|vayamos|id|vayan|fui|fuiste|fue|fuimos|fuisteis|fu
eron|fuera&fuese|fueras&fueses|fuera&fuese|fuéramos&fuése
mos|fuerais&fueseis|fueran&fuesen|fuere|fueres|fuere|fuéremo
s|fuereis|fueren|iré|irás|irá|iremos|iréis|irán|iría|irías|iría|iríamos
|iríais|irían|iba|ibas|iba|íbamos|ibais|iban|ir|yendo|ido|*|*|*")
```

Con los verbos modelo ingresados en la sección 4.1, en la mayoría de los casos no será necesario recurrir a esta plantilla.

4.2 Implementación del Analizador Morfológico de Verbos

La implementación del analizador morfológico de verbos se hizo en el lenguaje Python para aprovechar las ventajas mencionadas en la sección 3.2. El almacenamiento de la información ingresada en las plantillas de la sección 4.1 se hizo usando archivos planos de texto. El nombre de estos archivos es el mismo de las plantillas y supone un archivo por plantilla. Además, el diccionario de raíces también se maneja en un archivo de texto con este mismo nombre.
Los algoritmos de flexión y lematización emplean una clase llamada "gestorLexico", que permite manejar la información ingresada en el literal 4.1.
El método "buscConj" permite conjugar un verbo en infinitivo en un tiempo, persona y número en particular, de la manera siguiente:

```
def buscConj(self,verbInf,tmpo,pers,num):
"""almacena los resultados de la conjugación"""
  resultado={}
"""indexa  varias conjugaciones generadas"""
  index=0
"""obtiene los verbos Modelo asociados al verbo
infinitivo"""
  verbMods=self.gestorLexico.
  obtVerbModSegInf(verbInf)
```

```
"""si no hay verbos modelo asociados con el
infinitivo..."""
  if verbMods==None:
"""...busca en los verbos atípicos"""
    resultado=self.gestorLexico.
    obtConjVerbAtip
    (verbInf,tmpo,pers,num)
"""...si el infinitivo no existe en ningún
lado, retorna Null."""
    if len(resultado)==0:
      return None
    else: return resultado
"""para cada verbo modelo retornado (Se puede
dar que un verbo esté asociado con varios
verbos modelo)"""
  for i in verbMods:
"""verifica el participio irregular. Si existe
tal participio, almacena y no se conjuga como
el verbo modelo"""
#si se está conjugando en participio:
    if tmpo==10 and pers==3 and
       num==1:
      partIrr=self.gestorLexico.
      detPartIrr(i,verbInf)
#Si existe participio irregular:
      if not len(partIrr)==0:
        resultado[index]=partIrr
        index=index+1
        continue
"""obtiene el modelo de conjugación
correspondiente al tiempo, persona y número
deseado"""
    modConj=self.gestorLexico.
    obtModConjSegVerbMod
    (i,tmpo,pers,num)
"""obtiene las raíces correspondientes del
verbo a conjugar"""
    raices=self.gestorLexico.
    obtRaizSegInf(verbInf,i)
"""obtiene las terminaciones para el tiempo,
persona y número deseado, según el verbo
modelo"""
    terms=self.gestorLexico.
    obtTermSegVerbMod
    (i,tmpo,pers,num)
"""obtiene la información de verbo defectivo
para el verbInf según el verbo modelo."""
    infDef=self.gestorLexico.
    obtInfVerbDefSegVerbMod
    (i,verbInf)
"""realiza la conjugación. Aquí, se combina la
información de modConj con terms, que son
listas que pueden tener más de un elemento.
Además se tiene en cuenta si el verbo es
defectivo."""
    resultado[index]=self.
    combModConjYTerms
    (modConj,terms,raices,
    infDef,tmpo,pers,num)
    index=index+1
"""finaliza la conjugación"""
  return resultado
```

Un aspecto que se debe tener en cuenta en la conjugación son las combinaciones entre la raíz cero (0) y la terminación *dummy* ("*"). Su tratamiento es el siguiente:

- Si se presenta el caso raíz cero y terminación *dummy*, entonces la conjugación no existe.
- Si se presenta el caso raíz diferente a cero y terminación *dummy*, la conjugación es la raíz sola sin terminación.
- El otro caso que se puede presentar es raíz cero y terminación diferente a la *dummy*. Aunque en la conjugación se puede tratar, se prohíbe este caso en el léxico porque deteriora la eficiencia del proceso de lematización y, además, impone una restricción muy fuerte sobre el verbo modelo que la use y el conjunto de verbos que represente.

El método "lematVerb" permite lematizar un verbo conjugado a su forma de infinitivo, así:

```
def lematVerb(self,verbo):
  resultado={}
  for i in range(len(verbo)+1):
"""obtiene la raíz del paso i actual"""
    raiz=verbo[:len(verbo)-i]
"""obtiene la terminación del paso i"""
    term=verbo[len(verbo)-i:]
"""determina si la terminación no es vacía. Si
es vacía, el tratamiento de la raíz es
diferente."""
    if not term=="":
"""obtiene la información de la terminac"""
      infoTerm=self.gestorLex
      ico.obtInfTerm(term)
"""Si no se encontró la terminación, retorna al
primer for"""
      if infoTerm==None:continue
"""obtiene la información de la raíz"""
      infoRaiz=self.gestor
      Lexico.obtInfRaiz(raiz)
"""Si no se encontró la raíz, retorna al primer
for"""
      if infoRaiz==None:continue
"""realiza la lematización."""
      for j in infoTerm[term]:
#para cada raíz obtenida
        for k in infoRaiz[raiz]:
"""genera una conjugación a partir de infoTerm
e infoRaiz"""
          conjGenerada=self.
          buscConj(k[0],int(j[0]),
          int(j[1]),int(j[2]))
#si no existe la conjugación generada
          if conjGenerada==None:
"""sigue con la siguiente terminación"""
          continue
          for l ,m in conjGenerada.
            iteritems():
"""si el verbo ingresado es igual a algún verbo
generado..."""
            if verbo in m:
"""agrega al resultado el infinitivo que pasó
la conjugación con su información. También se
filtra la información para evitar resultados
repetidos"""
              if not resultado.
                has_key(k[0]):
                resultado[k[0]]=
                [self.generar
                Etiqueta(k[0],
```

```
                int(j[0]),
                int(j[1]),
                int(j[2]))]
              else:
                etiqueta=
                self.generar
                Etiqueta(k[0],
                int(j[0]),
                int(j[1]),
                int(j[2]))
#Se filtra la información
                if not etiqueta
                in resultado[k[0]]:
                  resultado[k[0]].
                  append(etiqueta)
    else:
"""Si la terminación es vacía, la raíz se debe
buscar en los verbos atípicos y también
lematizarla como raíz sin terminación."""
      infAtip=self.gestorLexico.
      busEnAtip(raiz)
      infRaiz=self.gestorLexico.
      busRaizSolaSinTerm(raiz)
"""agrega los resultados obtenidos de
busEnAtip"""
      for i,j in
        infAtip.iteritems():
        for k in j:
          if not resultado.
            has_key(i):
            resultado[i]=
            [self.generarEtiqueta
            (i,k[0],k[1],k[2])]
        else:
            resultado[i].append
           (self.generarEtiqueta
           (i,k[0],k[1],k[2]))
"""agrega los resultados obtenidos de
busRaizSolaSinTerm"""
      for i,j in
        infRaiz.iteritems():
        for k in j:
          if not resultado.
            has_key(i):
            resultado[i]=
            [self.generarEtiqueta
            (i,k[0],k[1],k[2])]
          else:
            resultado[i].append
            (self.generarEtiqueta
            (i,k[0],k[1],k[2]))
  return resultado
```

En el método "lematVerb", el método: "generarEtiqueta (verbInf, tmpo, pers y número)" transforma la información de la Tabla 1 al formato de la Tabla 3. Esto se hace porque en la lematización de verbos es común mostrar los resultados según esas etiquetas [14].

Tanto los métodos de conjugación como de lematización, se agruparon en una clase llamada analizadorMorfologico. Además, se construyó una interfaz gráfica de usuario para usar estos métodos.

4.3 Construcción del Diccionario de raíces

El diccionario de raíces se construyó usando un sistema auxiliar. Este sistema, a partir de un verbo del español y su correspondiente verbo modelo, genera todas las raíces correspondientes y las ingresa al diccionario de raíces; el proceso se puede repetir para cada verbo del español. Los verbos del español y su correspondiente verbo modelo se obtuvieron de los diccionarios existentes. Usando este sistema se procesaron aproximadamente 2300 verbos.

Tabla 3. Etiquetas para el proceso de lematización
Table 3. Tags for the Lemmatization process

Atributo	Valor	Código
TIPO	Principal (cualquier verbo)	M
	Semiauxiliar (verbo ser)	S
	Auxiliar (verbo haber)	A
MODO	Indicativo	I
	Subjuntivo	S
	Imperativo	M
	Infinitivo	N
	Gerundio	G
	Participio	P
TIEMPO	Presente	P
	Imperfecto	I
	Condicional	C
	Futuro	F
	Pasado	S
PERSONA	Primera	1
	Segunda	2
	Tercera	3
NUMERO	Singular	S
	Plural	P

4.4 Caso de estudio

En este caso de estudio se presenta el funcionamiento del Analizador Morfológico y la forma en que trabaja la conjugación y la lematización.

Con la definición de la raíz cero y la terminación *dummy*, se soluciona en parte el problema de los vacíos en la conjugación (inexistencia de una conjugación válida). En la Figura 3 se presenta cómo el verbo "embaír" aprovecha esta característica para generar en el tiempo Pretérito Indefinido las formas correctas.

Ahora, con la introducción, en el modelo de conjugación de la plantilla verbo modelo, del símbolo "[]" se solucionó el problema de la múltiple alternativa de conjugación. El verbo "yacer" aprovecha esta característica. En la Figura 4 se observa la conjugación del verbo en Presente Indicativo y el correcto reconocimiento de "yago", "yazco" y "yazgo" como formas conjugadas de este verbo.

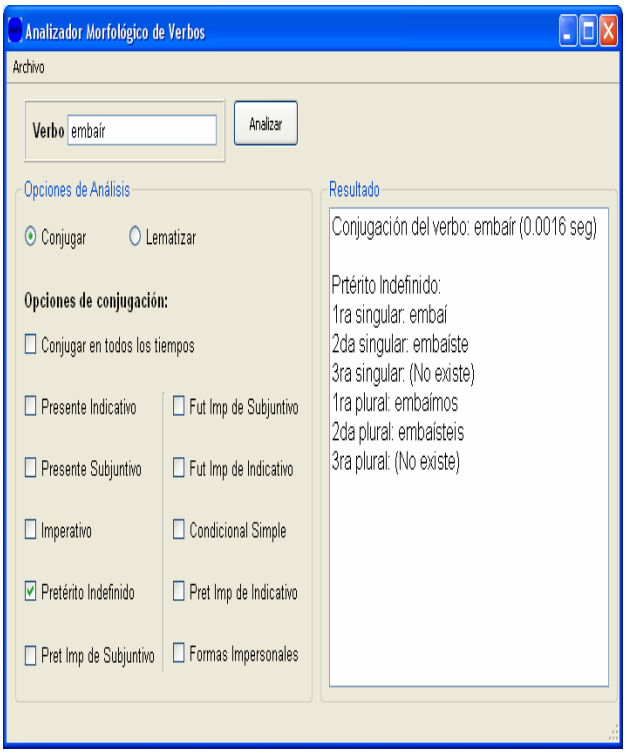

Figura 3. Correcta conjugación del verbo "Embaír"
Figure 3. Correct *Embaír* verb conjugation

5. CONCLUSIONES

Los analizadores morfológicos existentes para el español deben lidiar con las peculiaridades de este lenguaje y, aunque pueden generar cualquier conjugación de un verbo en particular, y asociar un verbo ya conjugado a su forma de infinitivo, estas propuestas poseen algunos problemas en el análisis, como el incorrecto manejo de los verbos defectivos y la incorrecta lematización de verbos que poseen dos o más alternativas de conjugación en un tiempo, persona y número en particular.

En este artículo se propuso un analizador morfológico de verbos del español, que aprovecha la existencia de los verbos modelo y la reutilización de terminaciones por parte de estos verbos. El analizador emplea una serie de plantillas que evitan el almacenamiento de todas las conjugaciones de un verbo en particular. Con estas plantillas es posible obtener los procesos de conjugación y lematización teniendo en cuenta las características de los verbos del español.

6. TRABAJO FUTURO

La propuesta se realizó usando archivos planos de texto para guardar las plantillas. El uso de estos archivos implica el tener que cargarlos a memoria cada vez que se desee utilizar el analizador morfológico. Esta solución, con el estado actual de los sistemas de gestión de bases de datos y las bases de datos empotradas, no es muy conveniente ya que el contenido de estos archivos es bastante grande, en especial el archivo que contiene el diccionario de raíces. Por esto, es necesario trasladar la información de estos archivos a algún mecanismo de bases de datos y modificar la implementación de los métodos de la clase "gestorLexico" para que accedan a esta información según el sistema de bases de datos elegido.

En este artículo se contempló el análisis morfológico de verbos del español. Sería interesante ampliar el alcance e incrementar el analizador para que reconozca cualquier palabra del español siguiendo el enfoque aquí descrito.

Después de obtener un analizador morfológico de palabras del español, es posible abordar el tema de la etiquetación de textos para así obtener una completa herramienta de análisis morfológico.

7. AGRADECIMIENTO

Este artículo se realizó en el marco de los siguientes Proyectos de Investigación: "Un modelo de diálogo para la generación automática de especificaciones en UN-Lencep", financiado por la DIME y "Definición de un Esquema Preconceptual para la Obtención Automática de Esquemas Conceptuales de UML", financiado por DINAIN y administrado por la DIME.

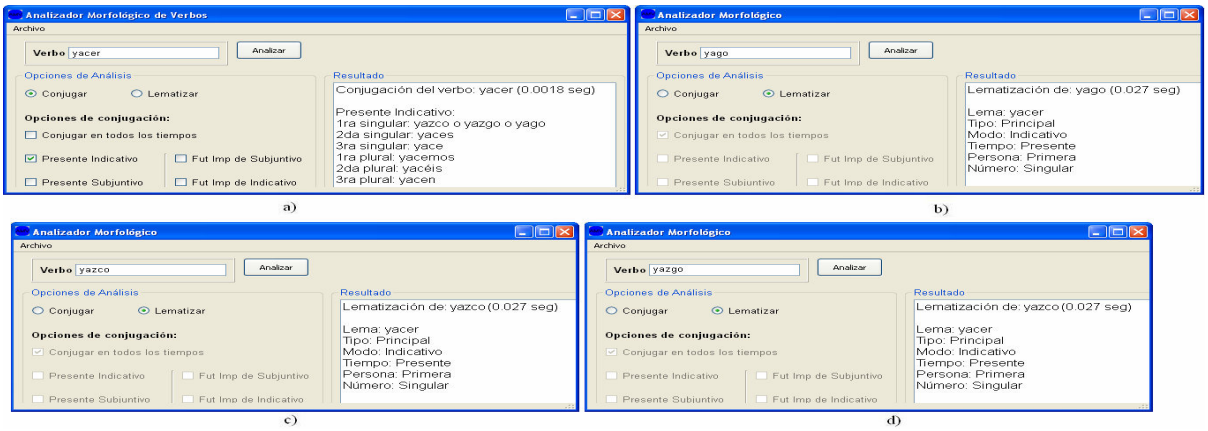

Figura 4. a) Conjugación del verbo "yacer" en presente indicativo. b), c) y d) Correcto reconocimiento de "yago", "yazco" y "yazgo"

Figure 4. a) *yacer* verb conjugation in present time. b), c) y d) Successful recognition of *yago, yazco* and *yazgo* verbs

REFERENCIAS

[1] MITKOV, R. The Oxford Handbook of Computational Linguistics. Reino Unido: Editorial Oxford University Press, 2003.

[2] ALONSO A. Lingüística. Ediciones Cátedra, 2002, 569p.

[3] VALENZUELA E. El verbo: 11.100 verbos conjugados cuarta edición. Editorial Bibliografía Internacional, 1990, 303p.

[4] GELBUKH A., SIDOROV G. Procesamiento automático del español con enfoque en recursos léxicos grandes; México DF, Instituto Politécnico Nacional, 2006, 240p.

[5] SANTANA, O.; PÉREZ, J.; HERNÁNDEZ, Z.; CARRERAS, F.; RODRÍGUEZ, G. FLAVER: Flexionador y lematizador automático de formas verbales. España, Revista Lingüística Española Actual XIX, 2, Ed. Arco/Libros 1997. 229-282p

[6] FLAVER. Flexionador y lematizador automático de formas verbales. Available: http://www.gedlc.ulpgc.es. [Citado 19 de Febrero de 2007].

[7] ZAMORANO, J, La morfología del español y la generación automática. España, Revista Procesamiento del lenguaje natural Vol.28, 2002, 35-43p.

[8] RAE. Real Academia Española. Available: http://www.rae.es. [Citado 19 de Febrero de 2007].

[9] AGME. Flexionador y lematizador automático de formas verbales. Available: http://www.gelbukh.com/agme/ http://www.cic.ipn.mx/~sidorov/agme/. [Citado 5 de Marzo de 2007].

[10] SERNA, M. Cómo conjugar correctamente los verbos. Editorial Idioma, 1991, 236p.

[11] Verbolog. Instituto de Verbología Hispánica. Available: http://www.verbolog.com. [Citado 19 de Febrero de 2007].

[12] GÓMEZ L. Manual del español correcto II: morfología y sintaxis, cuarta edición. Editorial Arco/Libros, 1993, 461p.

[13] MARZAL, A.; GRACIA I. Introducción a la programación con Python. Available: http://marmota.act.uji.es/MTP/pdf/python.pdf. [Citado 19 de Febrero de 2007].

[14] CIVIT, M. Criterios de etiquetación y desambiguación morfosintáctica de corpus en español. España, Revista Procesamiento del lenguaje natural, monografías 3, 2003, 364p.