Proyecto Ó penTrad: Traducción automática de código abierto para las lenguas del Estado español" (FIT-340101-2004-3, FIT-340001-2005-2)

Descripción del sistema de traducción es-eu Matxin.

(Entregables correspondientes a la documentación de los módulos 3 y 5 del proyecto)

Índice

1. Introducción

La presente *documentación*, redactada por el grupo Ixa, especifica la arquitectura general de la arquitectura *Matxin* y del sistema de traducción *es-eu* diseñados en el proyecto *OpenTrad* y la estructura y formato de cada uno de sus módulos. La estructura del sistema difiere sustancialmente de la arquitectura generada en el proyecto para las lenguas románicas (arquitectura *Apertium*), y es por ello que tiene una documentación, repositorio de software y nombre diferenciados. Por tanto, en este informe se denomina *Matxin* al sistema de traducción entre castellano y euskera y *Apertium* al sistema entre lenguas románicas. El conjunto del proyecto, por su parte recibe el nombre de *OpenTrad*. En cualquier caso, el sistema *Matxin* ha intentado reutilizar los módulos de *Apertium* en la medida de lo posible, y también *FreeLing (*sistema de código abierto para el análisis del castellano incorporado por la UPC).

Los módulos del proyecto inicial con los que se relaciona la presente *documentación* son el 1 (especificación), 3 (motor profundo), 5 (compiladores profundos) y también, en menor medida, los módulos 6 y 7 (datos monolingües y bilingües).

Este informe está basado en la documentación interna del proyecto, en comunicaciones a congresos que se han derivado del desarrollo del proyecto y en los sitios web de *apertium* y *FreeLing*: Esta documentación forma parte del sistema Matxin cuyo sitio web oficial está en la dirección http://matxin.sourceforge.net

A continuación se referencia documentación complementaria del proyecto:

1. **Especificación del formato de los recursos lingüísticos. Entregable 04-E01 (Informe interno *apertium*, http://apertium.sourceforge.net/extending.html**
2. Alegria I., Díaz de Ilarraza A., Labaka G., Lersundi M., Mayor A., Sarasola K., M. Forcada, S. Ortiz, L. Padró 2005
   An Open Architecture for Transfer-based Machine Translation between Spanish and Basque
   MT Summit. Workshop in Open Source MT

   http://ixa.si.ehu.es/Ixa/Argitalpenak/Artikuluak/index_html?Atala=Artikulua_Itzulpen_automatikoa
3. *Alegria I., Díaz de Ilarraza A., Labaka G., Lersundi M., Mayor A., Sarasola K. 2005*
   An FST grammar for verb chain transfer in a Spanish-Basque MT System
   FSMNLP Finite-State Methods and Natural Language Processing (poster session)

   http://ixa.si.ehu.es/Ixa/Argitalpenak/Artikuluak/index_html?Atala=Artikulua_Itzulpen_automatikoa
4. *Mayor A., Jordi Atseria , Eli Comelles 2005*
   TXALA un analizador libre de dependencias para el castellano
   XXI SEPLN (demo session)

   http://ixa.si.ehu.es/Ixa/Argitalpenak/Artikuluak/index_html?Atala=Artikulua_Itzulpen_automatikoa
5. *Carme Armentano-Oller, Antonio M. Corbí-Bellot, Mikel L. Forcada, Mireia Ginestí-Rosell, Boyan Bonev, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez, Felipe Sánchez-Martínez.*
   An open-source shallow-transfer machine translation toolbox: consequences of its release and availability.
   MT Summit. Workshop in Open Source MT
6. *Díaz de Ilarraza A., Mayor A., Sarasola K. 2000*
   Reusability of Wide-Coverage Linguistic Resources in the Construction of a Multilingual Machine Translation System
   MT 2000. University of Exeter, United Kingdom: 19-22 November 2000.

http://ixa.si.ehu.es/Ixa/Argitalpenak/Artikuluak/index_html?Atala=Artikulua_Itzulpen_automatikoa

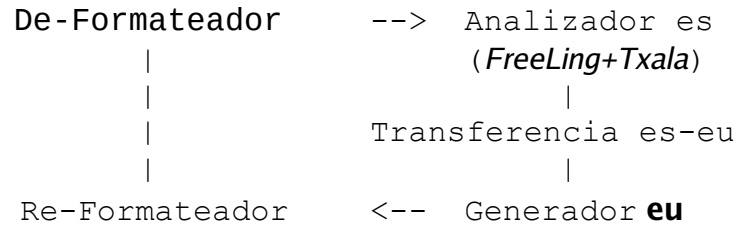
7. *Apertium:* http://apertium.sourceforge.net/
8. *FreeLing*: http://garraf.epsevg.upc.es/freeling/

El diseño consiste fundamentalmente en las tres etapas clásicas de los sistemas de transferencia (análisis, transferencia y generación) que se describen en la sección 2 de forma resumida y en la sección 4 más extendidamente. El formato de interacción entre los módulos está descrito en la sección 3, los formatos de los recursos lingüísticos en la sección 5 y el diseño de los programas en la sección 6. Finalmente, en los anexos se hace una descripción en profundidad de los recursos más complejos.

2. Arquitectura general del sistema

Los objetivos buscados son código abierto, interoperabilidad entre distintos sistemas y máxima convergencia con el desarrollo de *Apertium* y *Freeling*. Para ello, para el análisis del castellano se utiliza FreeLing (ya que da un análisis más profundo que el analizador de Apertium) y en la transferencia y generación se utilizan los transductores propuestos en *Apertium*.

El diseño se basa en la clásica arquitectura de transferencia, con tres componentes básicos: análisis es, transferencia es-eu y generación eu. Está basado en el trabajo previo del grupo IXA con su prototipo *Matxin* y en las aportaciones de diseño de *Apertium*. Se añaden dos módulos de *deformateado* y *reformateado* con el objetivo de mantener el formato de los documentos originales y permitir la navegación+traducción.



Según el diseño inicial no se hace ningún tipo de desambiguación semántica, pero el introducir en el léxico un importante número de entidades, colocaciones y otros términos multipalabra hace que esta limitación se vea acotada.

Con el diseño orientado a objetos en mente, se han definido tres objetos principales, que son *sentencia*, *chunk* y *nodo*. El *chunk* se puede asimilar al sintagma pero siempre en función de la salida del analizador, y el nodo a palabra, pero teniendo en cuenta que también hay unidades multipalabra.

A continuación hacemos una breve descripción de cada una de las etapas.

2.1. De-formateador y re-formateador

En principio se han previsto los formatos RTF y HTML.

Aunque se usa la misma tecnología usada en *Apertium*, la salida del *de-formateador* y la entrada del *re-formateador* varían pasando a ser dos ficheros en lugar de uno. Mientras en *Apertium* se introducen *superblancos* conteniendo información de formato, en *Matxin* se generan al de-formatear dos ficheros, uno con la información del formato y metadatos referenciando el texto, y

otro con el texto convertido a texto puro en un principio y a XML posteriormente, con la adición en este caso de información de ubicación. Esta información, que se detalla en el apartado 3, irá en los atributos *ord* (que expresa orden de los *chunks* en la sentencia y de los nodos en el *chunk*) y *alloc* (posición en texto analizado).

En el proceso de reformateado se pueden producir inconsistencias debido al cambio de orden de ciertos elementos en la frase traducida, siendo estas inconsistencias resueltas por medio de un heurístico sencillo.

2.2. El analizador

Ha sido desarrollado por la UPC y es una extensión de *FreeLing*, ya que se ha añadido al conjunto de módulos ya existente para el castellano (tokenizador, analizador morfológico, desambiguador y chunker) un analizador de dependencias.

El nuevo analizador de dependencias, que se ha denominado *Txala*, etiqueta la dependencia entre *nodos* dentro del *chunk* y entre *chunks* dentro de la *sentencia*. Esta información es obtenida en el formato de salida (ver sección 3) de forma indirecta, ya que en lugar de atributos específicos se expresa implícitamente por medio de la jerarquía de la etiqueta (por ejemplo, una estructura *node* dentro de otra quiere decir que la interna es dependiente de la externa).

Además de añadir esta funcionalidad, la salida del analizador se ha adaptado a la especificación del formato de intercambio que se describe en la sección 3.

2.2.1. Información obtenida

El resultado del análisis está compuesta por los tres elementos u objetos nombrados anteriormente:

- nodos: etiquetan las palabras o unidades multipalabra y proporcionan las siguientes informaciones: forma léxica, lema, parte de la oración (POS) e información morfológica de flexión.
- *chunks*: proporciona información del (pseudo)sintagma, tipo, información sintáctica y dependencia entre los nodos del mismo.
- sentencia: indica el tipo de sentencia y la dependencia entre los *chunks* de la misma.

2.2.2. Ejemplo

Para la siguiente frase

```
porque habré tenido que comer patatas
```

La salida estará compuesta por los siguientes chunks:

```
subordinate_conjunction: porque[cs]
verb_chain:
   haber[vaif1s]+tener[vmpp0sm]+que[cs] +comer[vmn]
noun_chain: patatas[ncfp]
```

2.3. La transferencia

Se mantienen los mismos tres objetos y formato de intercambio base del proceso: nodos, chunks

y sentencias. Los pasos que se siguen son los siguientes:

- transferencia léxica
- transferencia estructural dentro de la sentencia
- transferencia estructural dentro del chunk

2.3.1. Transferencia léxica

Primeramente se completa la transferencia léxica usando un diccionario bilingüe proporcionado por *Elhuyar*, que es compilado en un transductor léxico siguiendo las especificaciones y usando los programas de *Apertium*.

2.3.2. Transferencia estructural dentro de la sentencia

Debido a la diferente estructura sintáctica de las frases en cada idioma algunas informaciones son transferidas de unos *chunks* a otros, e incluso algunos *chunks* desaparecen o se crean.

En el ejemplo anterior, durante esta fase, la información de persona y número del objeto (tercera persona del plural) y el tipo de subordinación (causal) son importadas por la cadena verbal desde otros chunks .

2.3.3. Transferencia estructural dentro del *chunk*

Es un proceso complejo dentro de las cadenas verbales y más sencillo en las nominales. Una gramática de estados finitos (ver sección 4) ha sido desarrollada para la transferencia verbal.

En un principio se hizo el diseño para que esta gramática fuera compilada por los procesadores de *Apertium* o por medio del paquete de software libre FSA, pero esto ha resultado inabordable y se ha optado por una conversión de esa gramática a un conjunto de expresiones regulares que serán leídas y procesadas por un programa convencional que también tratará la transferencia de cadenas nominales.

2.3.4. Ejemplo

Para la frase ya comentada

```
porque habré tenido que comer patatas
```

La salida del análisis era:

```
subordinate_conjunction: porque[cs]
verb_chain:
   haber[vaif1s]+tener[vmpp0sm]+que[cs] +comer[vmn]
noun_chain: patatas[ncfp]
```

y la salida retocada de la etapa de transferencia será:

```
verb_chain:
   jan(main)[partPerf] / behar(per)[partPerf] / izan(dum)[partFut] /
                    edun(aux)[indPres][subj1s][obj3p]+lako[causal]
noun_chain: patata[noun]+[abs][pl]
```

2.4. La generación

También se mantienen los mismos objetos y formatos. Los pasos que se siguen son los siguientes:

- generación sintáctica
- generación morfológica

2.4.1. Generación sintáctica

Su principal labor es decidir el orden tanto de las palabras en el chunk como de los chunks en la frase.

El orden dentro del chunk se realiza por medio de una pequeña gramática que indica el orden de los elementos dentro de los sintagmas en euskera y que se expresa por un conjunto de expresiones regulares.

El orden de los chunks en la frase se decide por medio de un proceso recursivo basado en reglas.

2.4.2. Generación morfológica

Una vez decidido el orden la palabras dentro de cada *chunk* se debe proceder a la generación morfológica de la última palabra del *chunk* con la información morfológica propia o heredada en la fase de transferencia. Ello es debido a que en euskera, generalmente, la información de flexión (caso, número u otros atributos) se asigna al conjunto del sintagma, añadiéndola como sufijo al final de la última palabra del mismo. En las cadenas verbales, además de la última palabra, también se necesita generación morfológica adicional en otras palabras del sintagma.

Esta generación se lleva a cabo usando un diccionario morfológico generado por *IXA* a partir de su base de datos *EDBL*, que es compilado en un transductor léxico siguiendo las especificaciones, formatos y usando los programas de *Apertium*.

2.4.3. Ejemplo

Para la frase ya comentada el conjunto de pasos es el siguiente:

```
porque habré tenido que comer patatas
```

La salida del análisis era:

```
subordinate_conjunction: porque[cs]
verb_chain:
   haber[vaif1s]+tener[vmpp0sm]+que[cs] +comer[vmn]
noun_chain: patatas[ncfp]
```

La salida de la etapa de transferencia era:

```
verb_chain:
   jan(main)[partPerf] / behar(per)[partPerf] / izan(dum)[partFut] /
                    edun(aux)[indPres][subj1s][obj3p]+lako[causal]
noun_chain: patata[noun]+[abs][pl]
```

Tras la generación el resultado final será:

```
patatak jan behar izango ditudalako
```

Aunque los detalles de los módulos y los datos lingüísticos se presentan en la sección 4 hay que subrayar que el diseño es modular, estando organizado en los módulos básicos de análisis, transferencia y generación, teniendo diferenciados claramente los programas y los datos lingüísticos, y dentro de estos últimos los diccionarios y las gramáticas.

3. Formato de intercomunicación entre módulos

Un formato XML ha sido diseñado para intercomunicar las diversas etapas del proceso de traducción, habiéndose especificado todos los formatos en una única DTD. Se ha intentado diseñar un formato suficientemente potente para la aplicación de traducción, pero ligero para así permitir un tratamiento rápido.

Este formato va a facilitar la interoperabilidad (algún agente podría cambiar alguno de los módulos manteniendo los demás) y la adición de nuevos idiomas (en este caso la fase de transferencia deberá ser retocada).

Aunque la postedición de resultados no está recogida como objetivo del proyecto el formato ampliado también lo ha previsto (por medio de una etiqueta *ref*), con lo que se facilita el uso de estas herramientas en futuros proyectos.

El formato viene descrito por la siguiente DTD que puede ser utilizada para verificar la sintaxis de los formatos de intercambio:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT    SENTENCE     (CHUNK+)>
<!ATTLIST    SENTENCE
             ord          CDATA        #REQUIRED
             ref          CDATA        #REQUIRED
>
<!ELEMENT    CHUNK  (NODE, CHUNK*)>
<!ATTLIST    CHUNK
             ord          CDATA        #IMPLIED
             ref          CDATA        #IMPLIED
             length       CDATA        #IMPLIED
             type         (sn|grup-sp|grup-verb|conj-subord|F|...)  #REQUIRED
             si           (subj|obj|...) #IMPLIED
             mi           CDATA        #IMPLIED
             prep         CDATA        #IMPLIED
             cas          CDATA        #IMPLIED
             casref       CDATA        #IMPLIED
             casalloc     CDATA        #IMPLIED
             sub          CDATA        #IMPLIED
             subref       CDATA        #IMPLIED
             suballoc     CDATA        #IMPLIED
             rel          CDATA        #IMPLIED
             relref       CDATA        #IMPLIED
             relalloc     CDATA        #IMPLIED
             trans        CDATA        #IMPLIED
             subMi        CDATA        #IMPLIED
             objMi        CDATA        #IMPLIED
             datMi        CDATA        #IMPLIED
             headlem      CDATA        #IMPLIED
             headpos      CDATA        #IMPLIED
             headsem      CDATA        #IMPLIED
             leafpos      CDATA        #IMPLIED
>
<!ELEMENT    NODE   (NODE*)>
<!ATTLIST    NODE
             ord          CDATA        #IMPLIED
             form         CDATA        #IMPLIED
             lem          CDATA        #REQUIRED
             pos          CDATA        #IMPLIED
             mi           CDATA        #REQUIRED
             ref          CDATA        #IMPLIED
```

```
              alloc       CDATA       #REQUIRED
              sem         CDATA       #IMPLIED
              prep        CDATA       #IMPLIED
              cas         CDATA       #IMPLIED
              sub         CDATA       #IMPLIED
>
```

Como se puede ver aparecen los atributos *ord* y *alloc* que se usan para recuperar el formato y *ref* para la postedición. Los demás atributos corresponden a informaciones lingüísticas ya mencionadas.

Se ha preparado una hoja de estilo XSLT para ver gráficamente, en forma de árbol, la salida de cada uno de los módulos.

A continuación se describen tres ejemplos de la aplicación de este formato tras cada una de las fases básicas del proceso de traducción: análisis, transferencia y generación, siguiendo la traducción de la frase *"Un triple atentado sacude Bagdad"*

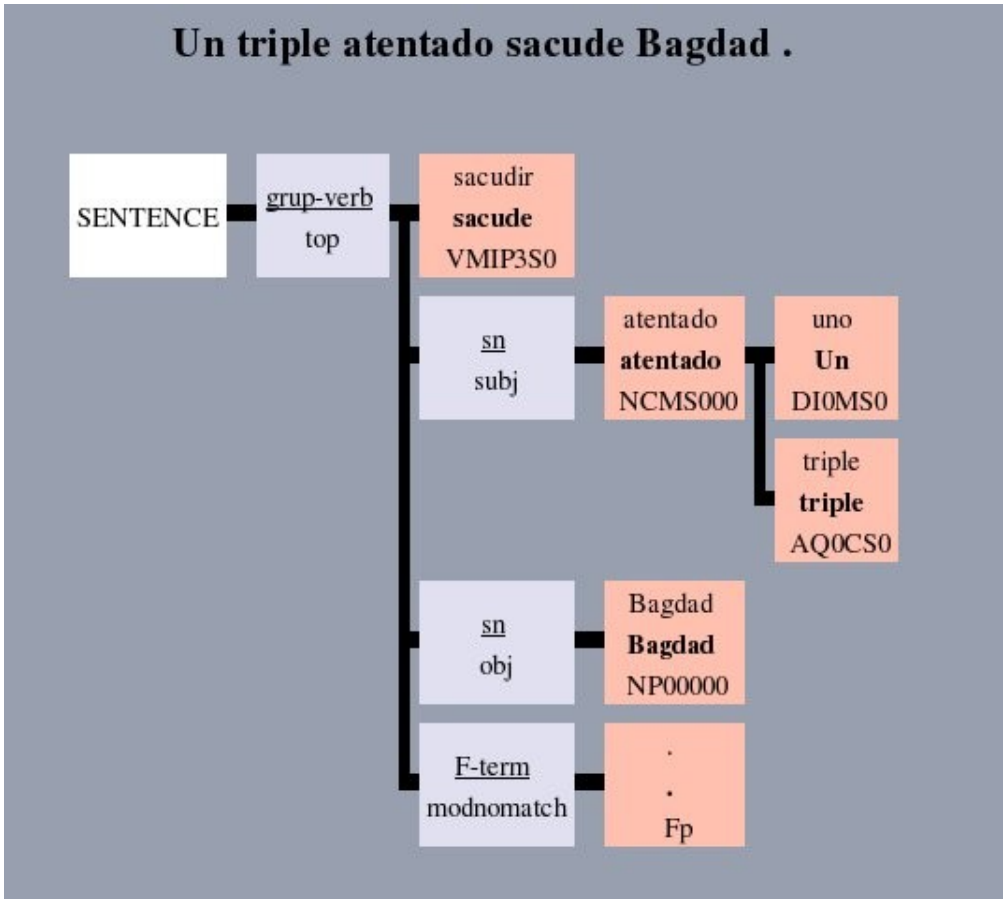
3.1. Formato tras el análisis

Se representa en el formato de intercambio como sigue:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<?xml-stylesheet type='text/xsl' href='profit.xsl'?>
<corpus>
<SENTENCE ord='1'>
<CHUNK ord='2' type='grup-verb' si='top'>
  <NODE ord='4' alloc='19' form='sacude' lem='sacudir' mi='VMIP3S0'> </NODE>
  <CHUNK ord='1' type='sn' si='subj'>
    <NODE ord='3' alloc='10' form='atentado' lem='atentado' mi='NCMS000'>
      <NODE ord='1' alloc='0' form='Un' lem='uno' mi='DI0MS0'> </NODE>
      <NODE ord='2' alloc='3' form='triple' lem='triple' mi='AQ0CS0'> </NODE>
    </NODE>
  </CHUNK>
  <CHUNK ord='3' type='sn' si='obj'>
    <NODE ord='5' alloc='26' form='Bagdad' lem='Bagdad' mi='NP00000'> </NODE>
  </CHUNK>
  <CHUNK ord='4' type='F-term' si='modnomatch'>
    <NODE ord='6' alloc='32' form='.' lem='.' mi='Fp'> </NODE>
  </CHUNK>
</CHUNK>
</SENTENCE>
</corpus>
```

La jerarquía de dependencia ya comentada se ha expresado más claramente por medio de tabulaciones, pero los programas la obtienen en función de las etiquetas *chunk* y *node* abiertas. Como se ve es un formato simple pero muy potente.

La misma información tratada con la referida hoja de estilo queda como sigue para la frase "*Un triple atentado sacude Bagdad*":



3.2. Formato tras la transferencia

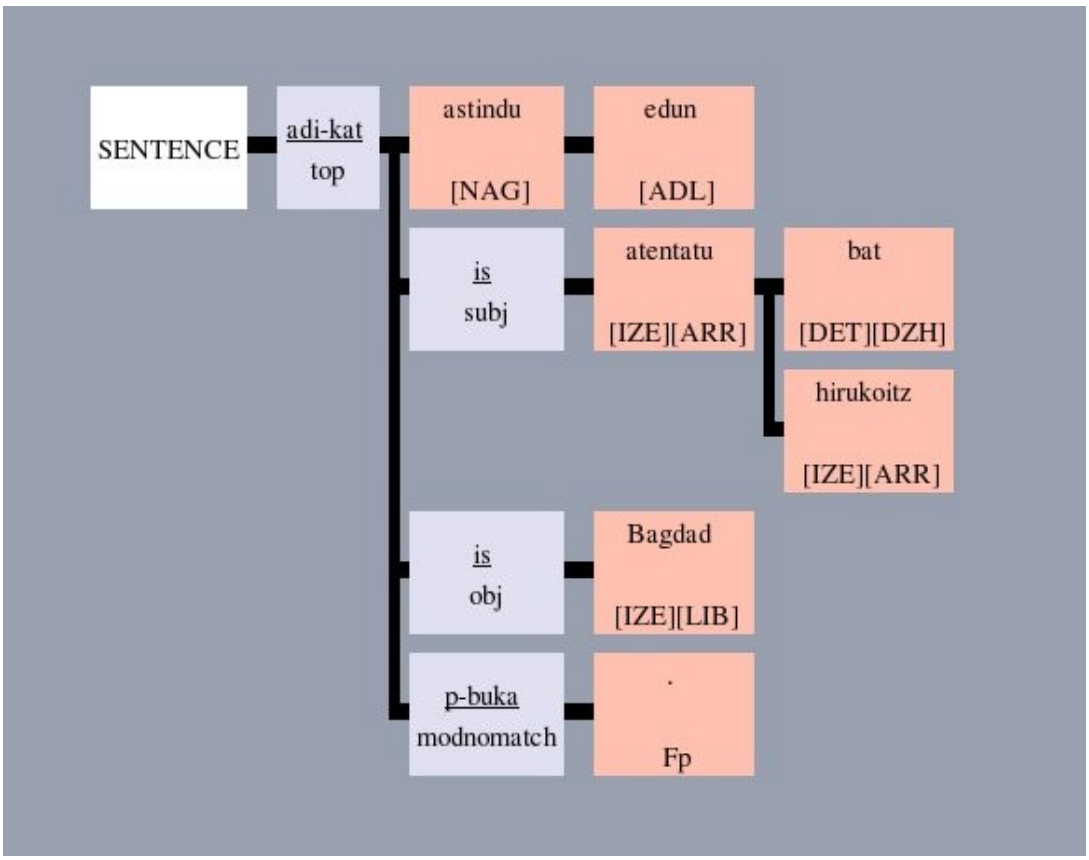
Se mantiene el formato, pero la información es traducida. Se añade el atributo *ref* para mantener la información de orden en la sentencia original. La postedición necesitará de la información de intercambio entre las distintas fases. Por otro lado la información correspondiente al atributo ord desaparece ya que un nuevo orden será calculado en la fase de generación.

También se puede observar que ciertos elementos han sido eliminados, pero su información ha sido heredada por otros.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<?xml-stylesheet type='text/xsl' href='profit.xsl'?>
<corpus>
<SENTENCE ref='1'>
  <CHUNK ref='2' type='adi-kat' si='top' headpos='[ADI][SIN]' headlem='_astindu_' trans='DU'
            objMi='[NUMS]' cas='[ABS]' length='2'>
    <NODE ref='4' alloc='19' lem='astindu' pos='[NAG]' mi='[ADI][SIN]+[AMM][ADOIN]+[ASP]
            [EZBU]'>
      <NODE ref='4' alloc='19' lem='edun' pos='[ADL]' mi='[ADL][A1][NR_HU][NK_HU]'/>
    </NODE>
    <CHUNK ref='1' type='is' si='subj' mi='[NUMS]' headpos='[IZE][ARR]' headlem='atentatu'
            cas='[ERG]' length='3'>
      <NODE ref='3' alloc='10' lem='atentatu' pos='[IZE][ARR]' mi='[NUMS]'>
        <NODE ref='1' alloc='0' lem='bat' pos='[DET][DZH]'> </NODE>
        <NODE ref='2' alloc='3' lem='hirukoitz' pos='[IZE][ARR]'></NODE>
      </NODE>
    </CHUNK>
```

```
    <CHUNK ref='3' type='is' si='obj' mi='[NUMS]' headpos='[IZE][LIB]' headlem='Bagdad'
             cas='[ABS]' length='1'>
      <NODE ref='5' alloc='26' lem='Bagdad' pos='[IZE][LIB]' mi='[NUMS]'> </NODE>
    </CHUNK>
    <CHUNK ref='4' type='p-buka' si='modnomatch' headpos='Fp' headlem='.' cas='[ZERO]'
             length='1'>
      <NODE ref='6' alloc='32' lem='.' pos='Fp'> </NODE>
    </CHUNK>
  </CHUNK>
</SENTENCE>
</corpus>
```

Visto con la hoja de estilo:



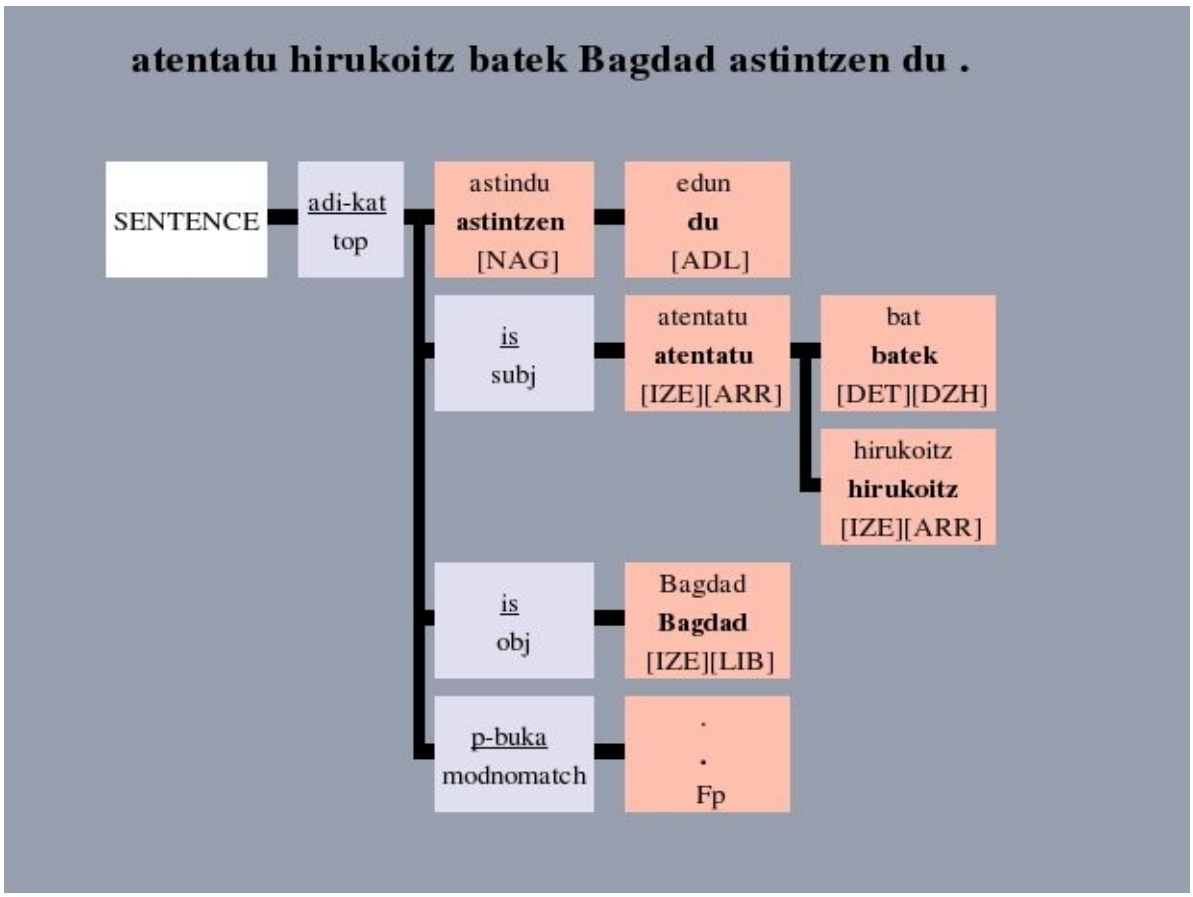
3.3. Formato tras generación

Los cambios más importantes son la reordenación por medio del valor recalculado para el atributo *ord*, y la generación morfológica de ciertos nodos (*edun* ->*ditudalako*, *patata* -> *patatak*).

```
<?xml version='1.0' encoding='iso-8859-1'?>
<?xml-stylesheet type='text/xsl' href='profit.xsl'?>
<corpus>
<SENTENCE ord='1' ref='1'>
  <CHUNK ord='2' ref='2' type='adi-kat' si='top' headpos='[ADI][SIN]' headlem='_astindu_'
            trans='DU' objMi='[NUMS]' cas='[ABS]' length='2'>
    <NODE form='astintzen' ord='0' ref='4' alloc='19' lem='astindu' pos='[NAG]' mi='[ADI][SIN]+
            [AMM][ADOIN]+[ASP][EZBU]'>
```

```
      <NODE form='du' ord='1' ref='4' alloc='19' lem='edun' pos='[ADL]' mi='[ADL][A1][NR_HU]
            [NK_HU]'/> </NODE>
    <CHUNK ord='0' ref='1' type='is' si='subj' mi='[NUMS]' headpos='[IZE][ARR]'
            headlem='atentatu' cas='[ERG]' length='3'>
      <NODE form='atentatu' ord='0' ref='3' alloc='10' lem='atentatu' pos='[IZE][ARR]'
            mi='[NUMS]'>
        <NODE form='batek' ord='2' ref='1' alloc='0' lem='bat' pos='[DET][DZH]'> </NODE>
        <NODE form='hirukoitz' ord='1' ref='2' alloc='3' lem='hirukoitz' pos='[IZE][ARR]'/>
      </NODE>
    </CHUNK>
    <CHUNK ord='1' ref='3' type='is' si='obj' mi='[NUMS]' headpos='[IZE][LIB]' headlem='Bagdad'
            cas='[ABS]' length='1'>
      <NODE form='Bagdad' ord='0' ref='5' alloc='26' lem='Bagdad' pos='[IZE][LIB]' mi='[NUMS]'>
            </NODE>
    </CHUNK>
    <CHUNK ord='3' ref='4' type='p-buka' si='modnomatch' headpos='Fp' headlem='.' cas='[ZERO]'
            length='1'>
      <NODE form='.' ord='0' ref='6' alloc='32' lem='.' pos='Fp'> </NODE>
    </CHUNK>
  </CHUNK>
</SENTENCE>
</corpus>
```

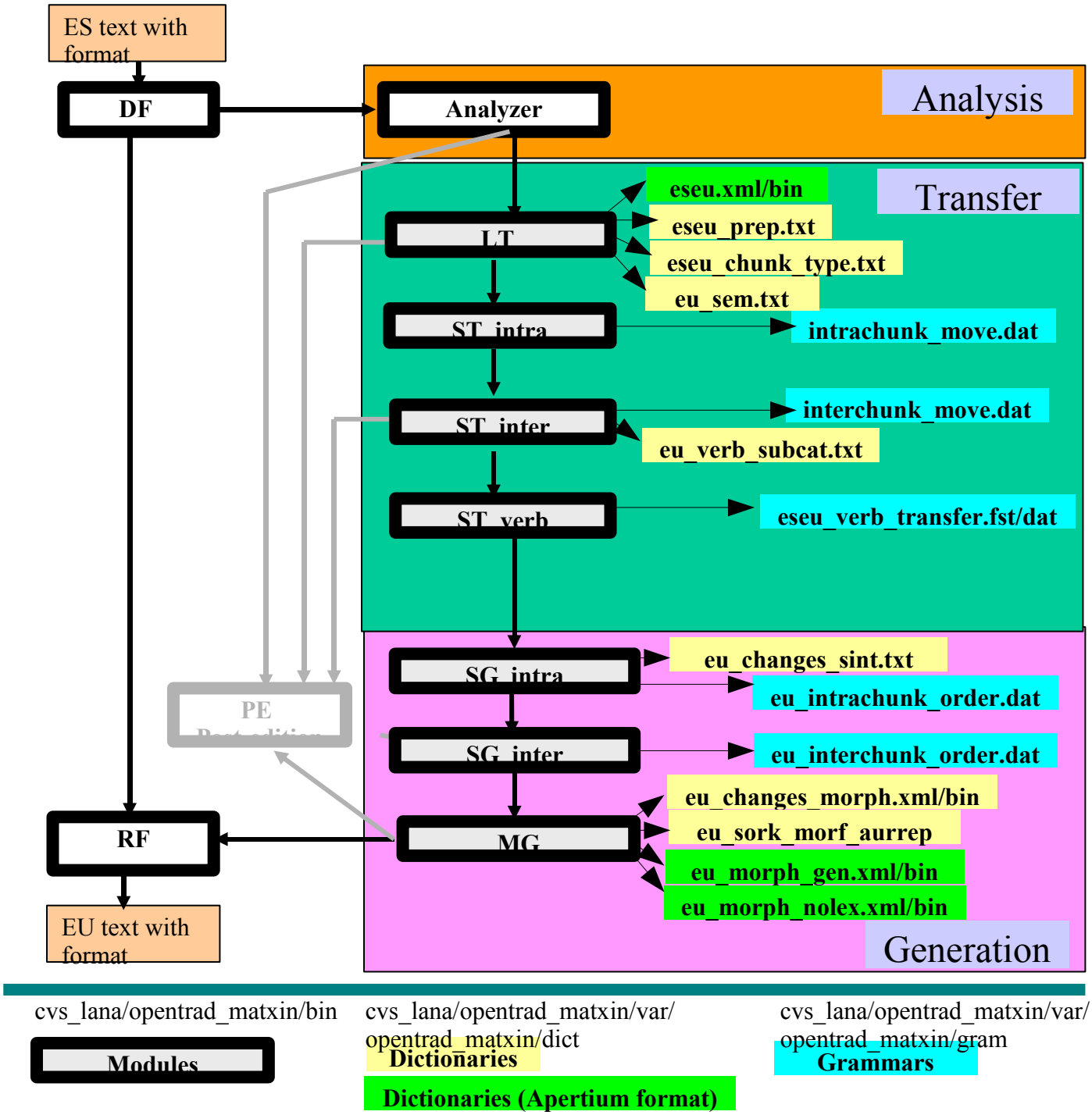
El resultado es la frase *"At entatu hirukoitz batek Bagad astintzen du"*



4. Arquitectura detallada: módulos y recursos lingüísticos

Hay que recordar que la parte del análisis reutiliza el paquete *FreeLing* y por lo tanto la documentación que se usará es la propia de ese proyecto: garraf.epsevg.upc.es/freeling/ y que también se usan funciones auxiliares del paquete *apertium* con el fin de generar y consultar los transductores y hacer el deformateado/reformateado de los textos: apertium.sourceforge.net

La arquitectura detallada de Matxin se puede observar en la figura 1.



La aplicación desarrollada reside en un servidor CVS, cuyo espejo está ubicado para su acceso público en el dominio *matxin.sourceforge.net*. Los elementos reflejados en la figura residen en cuatro subdirectorios:

- *src*: aquí se encuentran los programas fuente, que corresponden casi uno a uno a los nombres de los procesos de la figura 1.
- *var/matxin/dict*: diccionarios usados en la etapa de transferencia y en la de generación
- *var/matxin/gram*: gramáticas para transferencia y generación
- *bin*: programas ejecutables

Como puede observarse en la figura se distinguen por un lado claramente las tres fases de análisis, transferencia y generación, y por otro lado los módulos correspondientes a los programas, los diccionarios y las gramáticas; consiguiéndose así una arquitectura abierta y modular, que permite añadir nuevos idiomas sin necesitar cambios en los programas. Al ser código libre, será posible mejorar el sistema sin necesitar de modificar los programas, ya que se puede mejorar/ampliar los diccionarios y gramáticas sin necesidad de entender el código de los programas. Por supuesto también se puede modificar el código.

A continuación vamos a detallar los elementos de las distintas fases:

TRANSFERENCIA

LT: transferencia léxica
- diccionario bilingüe (compilado) (eseu.bin)
- diccionario de tipos de chunk (eseu_chunk_type.txt)
- diccionario de información semántica (eu_sem.txt)

ST_intra: transferencia sintáctica dentro del chunk
- gramática de cambio de información entre nodos (intrachunk_move.dat)

ST_inter: transferencia sintáctica entre chunks
- diccionario de subcategorización verbal (eu_verb_subcat.txt)
- diccionario de preposiciones (eseu_prep.txt)
- gramática de cambio de información entre chunks (interchunk_move.dat)

ST_verb: transferencia sintáctica del verbo
- gramática de transferencia verbales (compilada) (eseu_verb_transfer.dat)

GENERACIÓN

SG_intra: conversión y ordenación dentro del chunk
- dicc. de conversiones de informaciones sintácticas (eu_changes_sint.txt)
- gramática de ordenación dentro del chunk (eu_intrachunk_order.dat)

SG_inter: ordenación entre chunks
- gramática de ordenación entre chunks (eu_intrachunk_order.dat)

MG: generación morfológica
- dicc. de conversiones de información morfológica (eu_changes_morph.bin)
- diccionario de generación morfológica (compilado) (eu_morph_gen.bin)
- dicc de gen. morfológica para cualquier lema (compilado) (eu_morph_nolex.bin)
- dicc de gen. morfológica para medidas (compilado) (eu_measures_gen.bin)

gramatica de preprocesamiento morfológico (eu_morph_preproc.dat)
Finalmente, a continuación se listan los contenidos de los directorios comentados.

4.1. /var/matxin/dict

eseu.bin	eu_changes_morph.xml	eu_morph_nolex.xml
eseu.xml	eu_changes_sint.txt	eu_sem.txt
eseu_chunk_type.txt	eu_morph_gen.bin	eu_verb_subcat.txt
eseu_prep.txt	eu_morph_gen.xml	eu_measures_gen.xml
eu_changes_morph.bin	eu_morph_nolex.bin	eu_measures_gen.bin

4.2. /var/matxin/gram

eseu_verb_transfer.dat	eu_interchunk_order.dat	eu_morph_preproc.dat
eseu_verb_transfer.fst	eu_intrachunk_order.dat	interchunk_move.dat
intrachunk_move.dat		

4.3. src

Analyzer.C	config.h	ST_inter.C	XML_reader.h	reFormat.C
LT.C	README.txt	ST_intra.C	data_manager.h	simpleregex.C
MG.C	SG_inter.C	ST_verb.C	data_manager.C	simpleregex.h
Makefile	SG_intra.C	XML_reader.C		

A continuación se describen los recursos lingüísticos utilizados y posteriormente la estructura de los programas.

5. Formato de los recursos lingüísticos

Con el objetivo de impulsar una ingeniería de software apropiada, así como para facilitar la modificación del sistema de código abierto por lingüistas, se ha distribuido el conocimiento lingüístico en dos tipos de recursos (diccionarios y gramáticas) y se les ha dado el formato más abstracto y estandarizado que se ha podido.

Según lo descrito anteriormente los recursos lingüísticos fundamentales, exceptuando los de *FreeLing*, son los siguientes:

- Diccionarios:
  - Transferencia: diccionario bilingüe *es-eu*, diccionario de etiquetas sintácticas *es-eu*, diccionario semántico *eu*, diccionario de preposiciones *es-eu*, diccionario de subcategorización verbal *eu*
  - Generación: diccionario de cambios sintácticos, diccionario de cambios morfológicos y diccionarios morfológicos *eu*.
- Gramáticas:
  - Transferencia: gramática de transferencia de cadenas verbales *es-eu* y gramáticas de transferencia estructural *es-eu*.
  - Generación: gramática de preprocesado morfológico *eu*, gramática de ordenación intrachunk *eu* y gramática de ordenación interchunk *eu*.

En busca de una estandarización, los diccionarios bilingüe y morfológicos se especifican en el formato XML descrito para *apertium*, con lo que se hace compatible con este sistema.

Se ha intentado que las gramáticas sean de estados finitos, pero en el caso de la ordenación interchunk se ha optado por una gramática recursiva.

Se ha hecho un especial esfuerzo por optimizar la gramática de transferencia de cadenas verbales, ya que las transformaciones son profundas y podían llevar a ralentizar el sistema. A la hora de estandarizarla se ha optado por el lenguaje del paquete xfst (con algunas restricciones), bien documentado y muy potente, pero que tiene el problema de no ser software libre. Ante ello se ha optado por escribir un compilador que transforma esta gramática en un conjunto de expresiones regulares que se procesan en el módulo de transferencia.

El resto de los recursos lingüísticos son gramáticas que cubren diferentes objetivos y que tienen un formato específico. De momento tienen un formato no basado en XML[1], pero buscando un compromiso para que sea comprensible para los lingüistas y fácilmente tratable por los programas. En el futuro se diseñarán formatos y compiladores que hagan estas gramáticas más independientes de los programas.

En cualquier caso los datos lingüísticos están separados del programa para permitir a terceras partes modificar el comportamiento del traductor sin tener que cambiar el código.

5.1. Diccionario bilingüe es-eu `(eseu.xml)`

Sigue la especificación de *Apertium*. Se ha obtenido a partir del diccionario bilingüe de Elhuyar, y

1 El etiquetado XML de gramáticas no es habitual, aunque en Apertium sí se ha considerado

contiene los equivalentes en euskera para cada una de las entradas de castellano presentes en *FreeLing*. Este proceso se describe en profundidad en la referencia bibliográfica [6].

Debido a que no se hace desambiguación semántica solo se ha introducido la primera acepción de cada entrada (salvo en las preposiciones en las que se mantiene la ambigüedad), pero para paliar el problema se han añadido términos multipalabra obtenidos tanto del diccionario de Elhuyar como de un proceso de búsqueda automática de colocaciones.

En el anexo 2 se presentan el formato (que está exhaustivamente descrito en el capítulo 2 de la referencia [1]) y una pequeña parte del diccionario bilingüe.

5.2. Diccionario de etiquetas sintácticas es-eu (eseu_chunk_type.txt)

En este diccionario se encuentran las equivalencias entre las etiquetas sintácticas que proporciona el analizador *Freeling*, y aquellas etiquetas que serán usadas en la transferencia y generación en euskera. Es un diccionario muy simple que nos permite eliminar del código estas transformaciones.

Formato:

tipo_chunk_es tipo_chunk_eu #comentario

Ejemplo del contenido del diccionario:

```
sn          is          #sintagma nominal                      izen-sintagma
s-adj       adjs        #sintagma adjetivo                     adjektibo-sintagma
sp-de       post-izls   #sint. preposicional con la prep. "de" izenlagun-sintagma
grup-sp     post-sint   #sint. preposicional (excepto "de")    postposizio-sintagma
Fc          p-koma      #signo de puntuación "coma"            puntuazioa: koma
F-no-c      p-ez-koma   #signos de puntuación(excep. "coma")   koma es diren punt-ikurrak
número      zki         #cualquier número cardinal             edozein zenbaki kardinal
grup-verb   adi-kat     #grupo verbal                          aditz-katea
sadv        adbs        #sintagma adverbial                    adberbio-sintagma
neg         ez          #negación                              ezeztapena
coord       emen        #conjunción coordinada                 emendiozko juntagailua
```

5.3. Diccionario de preposiciones es-eu (eseu_prep.txt)

En este diccionario están las preposiciones del castellano con sus posibles equivalencias en euskera (un caso de declinación y a veces una posposición), la condición de selección (si la hay) de una de las posibles traducciones, y si después de mirar esas condiciones aún queda más de una posibilidad a elegir, una marca que señala qué equivalencias han de ser tenidas en cuenta en el proceso de selección con la información de subcategorización verbal.

Formato:

preposición_es caso_eu condicionSeleccion subcategorizacion

La condición de selección tiene el siguiente formato:

chunk-atributo='valor'

donde chunk puede ser "*my*" o "*parent*" según si se refiere a un atributo del propio chunk o a un atributo del chunk antecesor.

El campo *caso_eu* puede ser de dos tipos:

- un único caso de declinación
- un caso de declinación ++ una posposición / caso de declinación de la posposición

El caso de declinación de la posposición será usado en el proceso de selección con subcategorización verbal.

Ejemplo:

```
en          [INE]                       -                          +
a           +[AMM][ADIZE]+[DEK][ALA]    my.headpos='[ADI][SIN]'    -
a           [INE]                       my.headpos='[Zm]'          -
a           [DAT]                       -                          +
a           [ABS]                       -                          +
a           [ALA]                       my.si='cc'                 +
ante        [GEN]++aldean/INE           parent.headlem='izan'      -
ante        [GEN]++aurrean/INE          -                          +
ante        [GEN]++aurrera/ALA          -                          +
```

5.4. Diccionario de subcategorización verbal eu (eu_verb_subcat.txt)

Este diccionario es usado en el proceso de selección de la traducción correcta de las preposiciones de los complementos que acompañan a un verbo.

En el mismo se incluye la información de transitividad de cada verbo.

Formato

VerboEu transitividad1/casoSujeto1/casosComplementos2#transitividad2/casoSujeto2...

Las diferentes posibilidades de subcategorización aparecen ordenadas de mayor a menor según las frecuencias aparecidas en el corpus.

Para algunos verbos sólo se da la información de transitividad.

Ejemplo

```
aberastu          DU/ERG/ABS#DA/ABS/#DU/ERG/#DA/ABS/EN_BIDE#DU/ERG/ABS-INS#DU/ERG/ADJ#
     DA/ABS/ADJ#
abestu            DU/ERG/ABS#DU/ERG/#DU/ERG/ABS-INE#ZAIO/ABS/DAT#DU/ERG/INE#DA/ABS/#
     DU/ERG/ADJ#DU/ERG/EN_ARABERA#DU/ERG/AZ-INE#DU/ERG/ABS-INS#DU/ERG/INS#DA/ABS/INE#
     ZAIO/ABS/DAT-INS#ZAIO/ABS/DAT-INE#DA/ABS/INS#
abisatu           DIO/ERG/DAT#DU/ERG/#DIO/ERG/ABS-DAT#DU/ERG/KONP#DIO/ERG/DAT-INE#
     DU/ERG/INE#
absolbitu         DU/ERG/#DU/ERG/ABS#DU/ERG/ABL#DU/ERG/ABS-ALA#DU/ERG/ABS-ADJ#
     DU/ERG/ABS-MOT#DU/ERG/INS#
jario             ZAIO//#
jaulki            DU/ZAIO//#
jaundu            ZAIO//#
zuzenarazi        DA-DU//#
zuzeneratu        DA-DU//#
zuzperrarazi      DU//#
```

5.5. Diccionario semántico eu ***(eu_sem.txt)***

Para tomar ciertas decisiones en el proceso de traducción hay momentos en que es necesario tener información semántica sobre las palabras. Este diccionario contendrá información sobre rasgos semánticos de las palabras

Formato

nombreEu [rasgoSem Signo]

"rasgoSem" es la identificación de un rasgo semántico y Signo uno de los signos + ,- o ?. El símbolo ? " se usará cuando para algunas acepciones sea positivo y para otras negativo, o cuando no se ha podido establecer sus sentido.

De momento sólo se ha incluido la información sobre el rasgo semántico *animado/no animado*, si bien en el futuro se incluirán las que sean necesarias.

Ejemplo

```
abarka            [BIZ-]
abarkagile        [BIZ+]
abarketa          [BIZ-]
abarketari        [BIZ+]
abaro             [BIZ-]
abarrategi        [BIZ-]
abat              [BIZ?]
abata             [BIZ-]
abatari           [BIZ+]
```

5.6. Diccionario de cambios sintácticos ***(eu_changes_sint.txt)***

Para ordenar los nodos dentro de un chunk se necesita cierta información sintáctica sobre la categoría de los nodos. En el caso de los determinantes variará también según el lema del determinante (es decir un determinante de un tipo puede ir por delante del nombre, y otro del mismo tipo por detrás).

De momento en este diccionario se indica en la segunda columna qué determinantes se colocarán delante (*[DET][IZL]*) y cuales detrás ([DET][IZO]) en función del lema y la categoría, información especifica expresada en la primera columna.

Formato:

lema[infSintact] lema[infOrdenacion]

Ejemplo:

```
asko[DET][DZG]          asko[DET][IZO]
bana[DET][BAN]          bana[DET][IZO]
bat[DET][DZH]           bat[DET][IZO]
beste[DET][DZG]         beste[DET][IZL]
```

5.7. Diccionario de cambios morfológicos ***(eu_changes_morph.xml)***

Para generar la forma superficial de las palabras en euskera que hay que flexionar, es necesario conocer el lema, la categoría y la información del caso de declinación y su número. Pero a veces la categoría proveniente del diccionario bilingüe y que es usada para la ordenación de la palabra

dentro del chunk, no coincide con el conjunto de categorías usado en el diccionario morfológico. Por ello en esos casos se ha de conocer la información de categoría que servirá para su generación. Esta información se puede obtener usando el diccionario de cambios morfológicos que sigue el formato XML de *apertium*.

Ejemplo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE dictionary SYSTEM "http://www.torsimany.ua.es/esgleuca/dix.dtd">
<dictionary>
  <alphabet/>
  <sdefs/>
  <pardefs/>
  <section id="main" type="standard">
        <e><p><l>aarondar[IZE][ARR]</l><r>aarondar[ADJ][IZO]</r></p></e>
        <e><p><l>janari-denda[IZE][ARR]</l><r>denda[IZE][ARR]</r></p></e>
        <e><p><l>janari-saltzaile[IZE][ARR]</l><r>saltzaile[IZE][ARR]</r></p></e>
        <e><p><l>abade[ADJ][IZL]</l><r>abade[IZE][ARR]</r></p></e>
        <e><p><l>abadearen[ADJ][IZL]</l><r>abade[IZE][ARR]+[DEK][GEN]</r></p></e>
        <e><p><l>beira-ale[IZE][ARR]</l><r>ale[IZE][ARR]</r></p></e>
        <e><p><l>aitzindari izan[ADI][SIN]</l><r>izan[ADI][SIN]+[AMM][ADOIN]
            </r></p></e>
        <e><p><l>bandotan antolatu[ADI][SIN]</l><r>antolatu[ADI][SIN]+[AMM][PART]
            </r></p></e>
        <e><p><l>haize eman[ADI][SIN]</l><r>eman[ADI][SIN]+[AMM][ADOIN]</r></p></e>
        <e><p><l>haize emate[IZE][ARR]</l><r>eman[ADI][SIN]te[ATZ][IZE][ARR]
            </r></p></e>
```

5.8. Diccionario morfológico *(eu_morph_gen)*

La morfología en euskera es compleja y debido a su carácter aglutinante no se adapta a los estándares para el tratamiento morfológico en software libre como *ispell* o *aspell*. Cuando en el grupo IXA se diseñó el procesador morfológico se optó por la morfología de dos niveles que basándose en léxicos y reglas morfológicas permite la generación de un transductor para tanto análisis como síntesis.

El problema es que no existe software libre para compilar o transformar la morfología de 2 niveles, por lo que se ha optado por una transformación de los diccionarios y las reglas al formato propuesto en *Apertium,* eliminando los cambios fonológicos y transformándolos en paradigmas adicionales. Con ello se generan pseudo-lemas y pseudo-morfemas que no se corresponden con las formas canónicas.

Esto tiene un inconveniente, ya que el nuevo sistema es menos expresivo que el primitivo de dos niveles, además la transformación ha sido compleja y, en algunos casos aislados, ha perdido cobertura o ha sobregenerado. El principal inconveniente es que se ha limitado la capacidad de aglutinar morfemas en euskera, con lo que aunque para generación no va a resultar problemático, si lo será, en pequeña medida si se quiere usar el diccionario para análisis.

Además al ser generado automáticamente y usar formas no canónicas su legibilidad y comprensión por los lingüistas es menor.

El formato de *Apertium* es común con el del diccionario bilingüe y está exhaustivamente descrito en el capítulo 2 de la referencia [1]. Se compone de una sección para el alfabeto, otra para las etiquetas, una tercera para los paradigmas y una cuarta para pares de información léxica y

morfológica correspondientes a morfemas (lemas, prefijos o sufijos). En el anexo 3 se presentan una pequeña parte del diccionario morfológico. El fichero fuente es de tipo *xml* y el que usa el programa es de tipo *bin*.

5.9. Gramática de transferencia de cadenas verbales ***(eseu_verb_transfer.dat)***

Es una de las partes más compleja del sistema debido a que morfosintácticamente las cadenas verbales son muy diferentes en los dos idiomas. En la referencia [3] se hace una descripción somera del formato y función de las reglas que se aplican.

Por el momento el formato propuesto sigue la sintaxis de *xfst,* que aunque no es un estándar ni ofrece código abierto, si que es ampliamente conocido por su potencia, flexibilidad y documentación[2]. En un primer intento se pensó convertir la gramática al formato del software libre FSA[3], pero en las pruebas realizadas inicialmente hubo problemas de expresividad y finalmente los problemas de eficiencia nos hicieron desistir. Finalmente se ha construido un traductor **`fst2regex.pl`** de la gramática (basada en la sintaxis de xfst) **`eseu_verb_transfer.fst`** a un conjunto de expresiones regulares **`eseu_verb_transfer.dat`** que son leídas y procesadas por un módulo del sistema.

Las expresiones regulares serán aplicadas una a una, a una entrada que contiene la información del análisis de la cadena verbal en es, así como información de concordancia con los objetos e información de subordinación (si la hay). Las expresiones regulares van realizando modificaciones a esta entrada hasta conseguir una salida que contiene toda la información necesaria para generar la cadena verbal en eu.

5.9.1. Gramática

La gramática se compone de tres grupos de reglas:

- reglas de identificación y etiquetado
- reglas de conversión de atributos
- reglas de eliminación de información superflua

Las primeras son muy sencillas y etiquetan los distintos tipos de cadenas verbales identificados, seis en total. El formato genérico es el siguiente:

```
[ esVerbChainType @-> ... BORDER euVerbChainSchema ]
```

Siendo la parte de la izquierda una expresión regular que identifica el tipo de cadena verbal en castellano que se busca, y la parte de la derecha un conjunto de atributos correspondientes al esquema equivalente a ese tipo de verbo en euskera.

Las reglas de conversión de atributos que forman el segundo grupo tienen la siguiente estructura:

```
[ "euAttribute" @-> "euValue" || ?* esValues ?* BORDER ?* euValues ?* _ ]
```

que indica el valor que toma un atributo de un determinado tipo en el contexto especificado.

2 http://www.stanford.edu/~laurik/fsmbook/home.html
3 http://odur.let.rug.nl/~vannoord/Fsa/

La filosofía es que las primeras reglas añadan unos atributos abstractos, que estos atributos se vayan sustituyendo por valores concretos en función de los elementos de la cadena verbal en castellano, añadiéndose esta información a la derecha.

Finalmente las reglas de eliminación quitan la información de los atributos en castellano ya que no serán necesarios en adelante.

5.9.2. Ejemplo

Para la siguiente frase:

porque no habré tenido que comer patatas

La información proporcionada por el analizador sería la siguiente:

```
conjunction: porque[cs]
negative: no[rn]
verb_chain: haber[vaif1s]+tener[vmpp0sm]+que[cs]+comer[vmn]
noun_chain: patatas[ncfp]
```

que tras el proceso de trasferencia léxica quedaría en:

```
haber[vaif1s]+tener[vmpp]+que[cs]+comer[vmn]/[tr][3p][caus]/jan
```

Una regla del primer tipo (identificación y etiquetado) con la forma:

```
[ esVerbChainTypePerif1 @->... BORDER euVerbChainSchemaP1 ]
```

lo transforma en:

```
haber[vaif1s]+tener[vmpp]+que[cs]+comer[vmn]/[tr][3p][caus]/jan
     ==>P1> (main)Aspm/Per Aspp/Dum Aspd/Aux TenseM SubjObjDat +RelM
```

Una de las reglas que transforman los verbos perifrásticos tiene la forma siguiente:

```
[         [ "Aspp" @-> "[partFut]" || ?* [VMIF|VMIC|VAIC] ?* BORDER "P1" ?* _ ]
     .o. [ "Aspp" @-> "[partImp]" || ?* [VMIP|VMII] ?*
           BORDER "P1" ?*  [{hasi}|{amaitu}|{utzi}|{joan} ]  _  ]
     .o. [ "Aspp" @-> "[verbRad]" || ?* BORDER "P1" ?* {ari} ?*  _  ]
     .o. [ "Aspp" @-> "[partPerf]" || ?* BORDER "P1" ?*    ]
```

Transforman el contenido en:

```
haber[vaif1s]+tener[vmpp]+que[cs]+comer[vmn]/[tr][3p][caus]/jan
BORDER P1> (main)[partPerf] / behar(per)[partPerf] / izan(dum)[partFut]
     / edun(aux)[indPres][subj1s][obj3p]+lako[causal morpheme]
```

que tras la limpieza queda en:

```
jan(main)[partPerf] / behar(per)[partPerf] / izan(dum)[partFut]
     / edun(aux)[indPres][subj1s][obj3p]+lako[causal morpheme]
```

5.9.3. Expresiones regulares compiladas

El formato de las expresiones regulares obtenidas tras la compilación es el siguiente:

contextoALaIzquierda cadenaASustitir contextoALaDerecha cadenaQueSustituye

Ejemplo:

```
.*?\.querer\..*?              Per          .*?                    nahi_izan
.*?(\=\=\>)P1.*?              ari_izan     <PER>ADOIN.*?          aritu
.*?\.acabar\..*?\.de\..*?\/   Prt          .*?                    berri<PRT>[ADJ][IZO]
.*?dat2s.*?                   Ni           .*?                    [NI_ZU]
```

En la primera regla, por ejemplo, si encuentra la cadena "Pe r" teniendo a su izquierda la cadena ".querer.", sustituye "Pe r" por "h ahi_izan".

5.10. Gramática de cambio de información de los nodos al chunk (intrachunk_move.dat)

En la fase de transferencia es necesario pasar la información de algunos nodos al chunk al que pertenecen. Estos movimientos se definen en esta gramática.

Formato:

condicionNodo/atributoOrigen condicionChunk/atributoDestino modoEscritura

donde modoEscritura puede ser `overwrite, no-overwrite o concat.`

```
mi!=''      /mi         type!='adi-kat'   /mi         no-overwrite
mi='[MG]'   /mi         type!='adi-kat'   /mi         overwrite
prep!=''    /prep                         /prep       concat
prep!=''    /ref                          /casref     concat
prep!=''    /alloc                        /casalloc   concat
pos!=''     /pos                          /headpos    no-overwrite
lem!=''     /lem                          /headlem    no-overwrite
sem!=''     /sem                          /headsem    no-overwrite
pos='[Z]'   /pos                          /leafpos    overwrite
pos='[Z]'   /'[MG]'                       /mi         overwrite
pos='[Zu]'  /'[MG]'                       /mi         overwrite
```

Si se cumple la condición tanto en el nodo como en el chunk (en la condición se señala uno o varios atributos y el valor que deben tener, o no tener), el valor que se encuentre en el atributo origen es pasado al atributo destino del chunk en uno de los tres modos de escritura: *overwrite* (se escribe borrando lo que había anteriormente en el atributo), *no-overwrite* (se escribe sólo si el atributo no tenía ningún valor), o *concat* (se escribe añadiendo al valor que contenía previamente el atributo).

Por ejemplo, interpretando las dos primeras reglas del ejemplo, en un chunk verbal (adi-kat), se pasará la información morfológica (*mi*) del primero de los nodos que tengan alguna información morfológica al chunk. La información de los siguientes no se pasará, a no ser que contenga el valor '[MG]'.

5.11. Gramática de cambio de información entre chunks (interchunk_move.dat)

En la fase de transferencia es necesario pasar información de unos chunks a otros. Estos movimientos se definen en esta gramática.

Formato:

condOrigen/atribOrigen condDestino/atribDestino dirección modoEscritura

donde dirección puede ser down o up.

Ejemplo:

```
sub!=''&&type='adi-kat' /sub        si='obj'          /cas      down    overwrite
sub!=''                 /sub        type='adi-kat'    /rel      down    no_overwrite
sub!=''                 /subref     type='adi-kat'    /relref   down    concat
sub!=''                 /suballoc   type='adi-kat'    /relalloc down    concat
type='ez'               /'adi-kat-ez'  type='adi-kat' /type     up      overwrite
si='subj'               /mi         type='adi-kat'    /subMi    up      overwrite
si='obj'                /mi         type='adi-kat'    /objMi    up      overwrite
si='iobj'&&cas!='[DAT]' /mi         type='adi-kat'    /objMi    up      overwrite
```

Entre dos chunks se pasará la información del atributo origen al atributo destino, si se cumple la condición para ambos chunks (en la condición se señala uno o varios atributos y el valor que deben tener, o no tener) en uno de los tres modos de escritura: *overwrite* (se escribe borrando lo que había anteriormente en al atributo), *no-overwrite* (se escribe sólo si el atributo no tenía ningún valor), o *concat* (se escribe añadiendo al valor que contenía previamente el atributo), y en la dirección marcada: *down* (el chunk origen es el superior y el destino es hijo suyo) o *up* (el chunk origen es hijo del chunk superior al que se le pasa la información).

En la última regla del ejemplo se dice que si un chunk tiene información sintáctica (*si*) con el valor 'iobj' (objeto indirecto) y su caso de declinación (cas) es diferente a '[DAT]' (dativo), pasará al chunk que está encima suyo (dirección up) si este cumple la condición de que es de tipo verbal ('adi-kat'), la información de su atributo *mi* escribiéndola en modo *overwrite* en el atributo *objMi*.

5.12. Gramáticas de preprocesado morfológico eu (eu_morph_preproc.dat)

Para generar la forma superficial de las palabras en euskera que hay que flexionar, es necesaria la información del lema, la categoría y el caso de declinación y su número. Pero dependiendo de las categorías y casos esa información ha de ser ordenada de una u otra manera (ya que los morfemas no se encadenan siempre en el mismo orden) si se quiere obtener una generación morfológica correcta. Esta gramática define la ordenación de esta información.

Ejemplo:

```
[IZE][(IZB)](.*?)[(INE|ALA|ABL|ABU|GEL)]       LemaMorf +[DEK] Num +[DEK] Kas
[IZE][(IZB|LIB)]                               LemaMorf +[DEK] Kas [MG]
...
[DET][ERKARR][NUMS](.*?)[(ABS|SOZ|DES|GEN)]    LemaMorf +[DEK] Kas
[DET][ERKARR][NUMS](.*?)[(ERG|DAT)]            LemaMorf +[DEK] Kas Num
[DET][ERKARR][NUMS](.*?)[(INE|ABL)]            LemaMorf +[DEK] [MG] +[DEK] Kas
[DET][ERKARR][NUMS](.*?)[(.*?)]                LemaMorf +[DEK] Num +[DEK] Kas
...
[DET][ERKARR](.*?)[(SOZ|DES|GEN)]              LemaMorf +[DEK] Kas
[DET][ERKARR](.*?)[(ABS|ERG|DAT|INS)]          LemaMorf +[DEK] Kas Num
[DET][ERKARR](.*?)[(.*?)]                      LemaMorf +[DEK] Num +[DEK] Kas
...
gutxi[DET][DZH](.*?)                           LemaMorf +[DEK] Kas
```

```
...
(.*?)                                          LemaMorf +[DEK] Num +[DEK] Kas
```

En la columna de la izquierda aparece una expresión regular donde puede aparecer el lema, la información sintáctica y morfológica y el caso de declinación de la palabra que se va a declinar. Según esa información se ordenarán los morfemas en el orden indicado en la segunda columna. "lemaMorf" expresa el lema y la información sintáctica, "Num" expresa la información de número y "Kas" la de caso de declinación.

Así podemos ver que para generar las formas declinadas de los determinantes demostrativos en singular "`[DET][ERKARR][NUMS])` "hau", "hori" y "hura", según los casos de declinación se ordenará de manera diferente que los que son en plural "hauek", horiek, "haiek". .

En algunos casos, como para "gutxi", esa palabra se ordena de manera diferente que el resto de palabras de esa categoría.

En la última línea se define el orden de los morfemas por defecto.

5.13. Gramáticas de ordenación intrachunk eu (eu_intrachunk_order.dat)

Los nodos dentro de un chunk se ordenarán en euskera siguiendo el patrón que marca esta gramática.

Formato

```
tipoChunk   (infSintact)...(infSintact)([BURUA])(infSintact)...(infSintact)
```

[BURUA] será el lugar donde se coloque el nodo raíz del chunk. El resto de nodos se colocarán en el lugar correspondiente a su información sintáctica.

Ejemplo

```
is
([DET][IZL])([DET][IZL])([DET][IZL])([DET][IZL])([DET][ORD])([Z])([ADB][ADOARR])([ADJ]
[IZL])([ADJ][IZL])([ADJ][IZL])([ADJ][IZL])([IZE][IZB])([IZE][IZB])([IZE][IZB])([IZE][I
ZB])([IZE][ARR])([IZE][ARR])([BURUA])([ADJ][IZO])([ADJ][IZO])([ADJ][IZO])([LOT][JNT])(
[ADJ][IZO])([DET][IZO])([DET][IZO])([DET][IZO])([DET][IZO])

adi-kat ([ADB])([BURUA])([PER])([PRT])([ADM])([ADL])

adi-kat-ez      ([PRT])([ADL])([ADB])([BURUA])([PER])([ADM])
```

Por ejemplo, en el caso de las cadenas verbales en oraciones afirmativas (*adi-kat*), se ordenará poniendo antes del nodo raíz, el nodo que contienen un adverbio ([ADB]) si es que existe, y detrás el verbo perifrástico ([PER]), la partícula verbal ([PRT]), el verbo modal ([ADM]) y el verbo auxiliar ([ADL]), en ese orden y en el caso de que existan cada uno de ellos.

Podemos ver que, en cambio, en las cadenas verbales en oraciones negativas (adi-kat-ez) el orden cambia, colocándose delante la partícula verbal ([PRT]), el verbo auxiliar ([ADL]) y el adverbio ([ADB]), y detrás el verbo perifrástico ([PER]) y el verbo modal ([ADM]), siempre que existan.

Para la traducción de "he tenido que venir" y "no he tenido que venir", se ordenarían de esta manera:

```
adi-kat     >>    etorri      behar       izan         dut
                  [BURUA]     [PER]       [ADM]       [ADL]
adi-kat-ez  >>    dut          etorri     behar       izan
                  [ADL]       [BURUA]     [PER]       [ADM]
```

5.14. Gramáticas de ordenación interchunk eu ***(eu_interchunk_order.dat)***

El orden de los chunks dentro de la frase se decidirá siguiendo un proceso recursivo, que recorre el árbol en postorden, ordenando cada chunk con cada uno de los chunks que cuelgan de él.

Formato

tipoChunkSup tipoChunkInf posiciónRelativa orden

Dónde *orden* puede especificarse de la siguiente forma:

x1.x2	el chunk inferior (x2) se coloca detrás del último chunk ordenado hasta ahora - - x1 - - => - - x1 - - x2
x2.x1	el chunk inferior (x2) se coloca inmediatamente delante del chunk superior (x2) - - x1 - - => - - x2 x1 - -
x2+x1	el chunk inferior (x2) se coloca inmediatamente delante del chunk superior (x2) no permitiendo que ningún otro chunk se sitúe entre ellos. Si algún otro chunk se ha de ordenar por delante (con el orden x2.x1 o x2+x1), se situará inmediatamente delante de este chunk que acabamos de ordenar. - - x1 - - => - - [x2 x1] - -

Y *posiciónRelativa* señala en que posición se encontraba el chunk inferior con respecto al chunk superior, en la oración en la lengua origen.

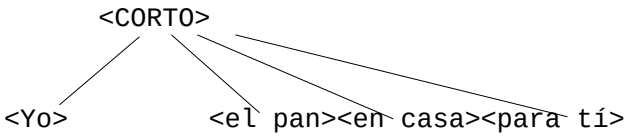
Ejemplo:

```
adi-kat-ez   ez         .*?         x2+x1
adi-kat      .*?        >1          x1.x2
adi-kat      .*?        =1          x2.x1
adi-kat      .*?        <1          x2.x1
.*?          .*?       .*?         x2.x1
```

Por ejemplo cuando un chunk de tipo adi-kat-ez (grupo verbal) tiene un chunk hijo de tipo 'ez' (partícula de negación equivalente a 'no'), esa partícula se colocará inmediatamente delante de la cadena verbal y ningún otro chunk se podrá situar entre ellas.

En cambio los chunks que dependan de un chunk de tipo *adi-kat* (grupo verbal), se colocarán delante, si en castellano se encontraban delante del verbo (posición relativa <1), o inmediatamente después de él (pos. rel. =1). Los chunks que se encontraban en castellano más a la derecha, en euskera también se colocarán detrás, quedando uno tras otro en el mismo orden que estaban en castellano.

Por ejemplo, en la frase "*Y o corto el pan en casa para tí*"



pos.relativa	-1	0	1	2	3
orden	x2.x1		x2.x1	x1.x2	x1.x2

se ordenará de esta manera

<Nik><ogia><MOZTEN DUT><etxean><zuretzat>
(yo) (el pan) (corto) (en casa) (para tí)

6. Diseño de los programas

En el capítulo 2 se ha descrito brevemente el esquema general y en el capítulo 4 los módulos que componen el programa. En este capítulo se quiere exponer más detalladamente, y desde un prisma más propio de desarrollo de programas, los componentes del sistema de traducción

El analizador, el módulo de transferencia y el módulo de generación son los 3 componentes fundamentales del sistema (ver figura 1). Estos módulos se comunican por medio de la estructura XML definida en el capítulo 3.

El primer elemento, el analizador, ha sido desarrollado por la UPC y se distribuye y documenta independientemente en el paquete *FreeLing (*http://garraf.epsevg.upc.es/freeling/).

Los módulos de transferencia y generación tienen un diseño y programación coherente que hacen uso, cuando es adecuado, de los módulos básicos del paquete *apertium (*http://apertium.sourceforge.net/).

Los programas que se refieren a continuación están disponibles bajo licencia GPL en el paquete *Matxin* (http://matxin.sourceforge.net/)

6.1. Metodología y orientación a objetos

Como se ha dicho se ha realizado un diseño orientado a objeto, donde los objetos básicos son los elementos del formato de intercambio: nodo (palabra/lema), *chunk* (pseudo-sintagmas) y sentencia. Como se puede observar en los ejemplos del capítulo 4, con estos elementos se representa el árbol de dependencias de la frase dividida en *chunks* y las dependencias entre palabras/lemas de cada chunk, además de la información propia de cada palabra/lema.

Para el ejemplo ya comentado

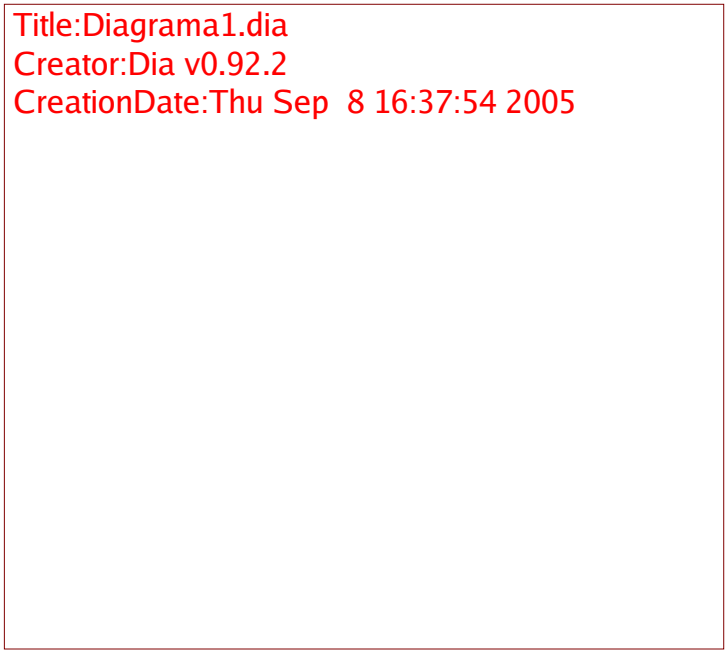
`porque habré tenido que comer patatas`

La información proporcionada por el analizador sería la siguiente:

- una sentencia
- tres *chunks*: *porque*, *habré_tenido_que_comer* y *patatas,* en ese orden de dependencia, dependiendo el segundo del primero y el tercero del segundo.
- El segundo *chunk* se divide en cuatro nodos, siendo *comer* el nodo raíz y dependiendo de él los otros tres.

Esto representa un árbol (que será la representación interna que se use) de la forma siguiente:

La representación entre módulos verdaderamente queda como sigue:




```
<SENTENCE ord="1">
  <CHUNK ord="1" type="conj-subord">
    <NODE ord="1" form="porque" lem="porque" mi="CS" alloc="1"/>
    <CHUNK ord="2" type="grup-verb">
      <NODE ord="4" form="comer" lem="comer" mi="VMN0000" alloc="25">
        <NODE ord="1" form="habré" lem="haber" mi="VAIF1S0" alloc="8"/>
        <NODE ord="2" form="tenido" lem="tener" mi="VMP00SM" alloc="14"/>
        <NODE ord="3" form="que" lem="que" mi="CS" alloc="21"/>
      </NODE>
      <CHUNK ord="3" type="sn" si="obj">
        <NODE ord="1" form="patatas" lem="patata" mi="NCFP000" alloc="31"/>
      </CHUNK>
    </CHUNK>
  </CHUNK>
</SENTENCE>
```

Tanto en la transferencia como en la generación se necesitan varias etapas. En transferencia se distinguen la léxica y la estructural y en generación la sintáctica y la morfológica. La transferencia estructural y la generación sintáctica se dividen, a su vez, en subfases. Llamaremos *intra* al tratamiento que procesa los nodos dentro de un *chunk*, e *inter* al tratamiento que procesa *chunks* dentro de la frase.

Siempre que ha sido factible se ha independizado el código del conocimiento lingüístico, recayendo este ultimo en gramáticas, diccionarios y otros elementos descritos en el capítulo 4.

A continuación se describen los módulos más detalladamente.

6.2. Módulos de transferencia

El proceso de transferencia es bastante complejo y tiene dos módulos básicos:

- Transferencia léxica
- Transferencia estructural

La transferencia estructural tiene módulos *intra* e *inter*, que se describen más adelante.

6.2.1. Transferencia léxica

Se basa en el diccionario bilingüe descrito en el apartado 4.1. Se buscan en el diccionario todos los lemas provenientes del castellano salvo los nodos verbales que no son raíces (verbos auxiliares y componentes de verbos perifrásticos). Estos últimos se marcan y se dejan para transformarlos en la transferencia estructural.

A continuación el pseudo-algoritmo:

```
conversion de atributo ord a ref  (sentence/chunk/node)
for (chunks)
      if (chunk verbal)
            for (nodos)
                  if (nodo raiz chunk)
                        consulta_diccionario_y_lem_pos (nodo)
                  else no_transferencia_marcado_lema
      else
            consulta_diccionario_y_lem_pos_im (nodo)
            consulta_semantica (nodo)
```

6.2.2. Transferencia estructural

Se realiza realizan transferencias de atributos entre nodos y chunks y entre distintos chunks para conseguir una estructura equivalente en la lengua meta.

Tranferencia estructural intra

El objeto a tratar es el chunk. En general se traspasan atributos de nodos al chunk al que pertenecen, siguiendo lo especificado en la gramática de cambio de información. En los casos en los que tras la transferencia léxica un nodo se ha quedado sin lema, este nodo desaparece.

A continuación el pseudo-algoritmo:

```
for (chunks)
      for (nodos)
            subir_atributos_(gramatica,nodos,chunk)
            if (lema_vacio)
                  eliminar_nodo(nodo)
```

Transferencia estructural inter

Se realiza sobre el objeto sentencia. Se traspasan atributos de un chunk a otro según lo especificado en la gramática de cambio de información. Además, se encarga de resolver los casos de los sintagmas nominales (chunks) en función del tipo de verbo principal y preposiciones del resto de los chunks; además decide la transitividad del verbo en eu.

A continuación el pseudo-algoritmo:

```
for (chunk)
      bajar_atributos(chunk_actual, chuk_dependientes, gramatica)
      traducir preposiciones(chunk_dependientes)
      if (chunk verbal)
            subcategorizacion(chunk_actual, chuk_dependientes)
            asignar_transitividad(chunk_actual)
      subir_atributos(chunk_actual, chunk_ascendentes)
```

```
      if (chunk_vacio)
            eliminar_chunk
```

Transferencia estructural verb

Utiliza directamente la gramática de transferencia de cadenas verbales descrita en el apartado 4.3

A continuación el pseudo-algoritmo:

```
for (chunks_verbales)
      aplicar gramatica_transferencia(chunk)
```

6.3. módulos de generación

Como en la transferencia hay dos pasos principales pero en el orden inverso; primero la generación sintáctica (que como la transferencia estructural funciona a nivel de chunk y sentencia) y posteriormente la generación morfológica (que como la transferencia léxica trata los nodos, en este caso no todos)

6.3.1. Generación sintáctica

Tiene una primera parte de reordenación de los chunks dentro de la sentencia (*inter*), y otra posterior de reordenación de los nodos en los chunks (*intra*).

Generación sintáctica inter

Trabaja de forma recursiva y sigue la gramática de ordenación interchunk. Se recorre el árbol en postorden y se decide el orden relativo de un chunk y su hijo, según las reglas definidas en la gramática

A continuación el pseudo-algoritmo:

```
for (ordenar(chunk, post-order))
      definir_orden(parent, child)
```

Generación sintáctica intra

Ordena los nodos dentro de los chunks según una gramática de expresiones regulares. Este elemento se describe en el apartado 5.13. La gramática está en var/matxin/gram/eu_intrachunk_order.dat.

```
For (chunk)
      definir_orden(nodes)
```

6.3.2. Generación morfológica

Finalmente se transfiere la información morfológica del *chunk* al nodo correspondiente a la ultima palabra para hacer uso de esta información en la siguiente etapa.

Se basa en el diccionario morfológico del apartado 4.2. Adicionalmente existe dos tabla hash con los objetivos de completar la información morfológica y ordenarla en *var/matxin/dict/eu_changes_morph.bin* y var*/matxin/dict/eu_changes_sint.txt*

A continuación el pseudo-algoritmo:

```
for (chunks)
```

```
nodo=(bilatu_azken_hitza)
egokitu_lema_pos(nodo)
kontsultatu_sorkuntzako_hash(nodo)
kontsultatu_ordenaketa_hash(nodo)
sortu_forma(nodo)
```

7. Anexo 1: Información morfológica en euskera de los diccionarios

7.1. Categorías léxicas (15)

7.1.1. Categorías principales (10)

- **ADB** — **ADVERBIO**
  - **ARR** — **COMÚN** — gaur
  - **GAL** — **INTERROGATIVO** — noiz ('cuándo')
- **ADI** — **VERBO**
  - **SIN** — **SIMPLE** — ekarri
  - **ADK** — **COMPUESTO** — lo egin
  - **ADP** — **PERIFRÁSTICO** — ahal izan
- **ADJ** — **ADJETIVO**
  - **ARR** — **COMÚN** — handi
  - **GAL** — **INTERROGATIVO** — nongo (‘de dónde’)
- **BST** — **OTROS** — baldin ('si')
- **DET** — **DETERMINANTE**
  - *ERK* — *DEMOSTRATIVO*
    - **ERKARR** — **COMÚN** — hau
    - **ERKIND** — **INTENSIFICADO** — berau ('esto mismo')
  - *NOL* — *MODAL*
    - **NOLARR** — **COMÚN** — edozein
    - **NOLGAL** — **INTERROGATIVO** — zein
  - *ZNB* — *NUMERAL*
    - **DZH** — **DEFINIDO** — bi
      - **BAN** — **DISTRIBUTIVO** — bina ('dos a cada')
      - **ORD** — **ORDINAL** — bigarren
    - **DZG** — **INDEFINIDO** — zenbait ('algunos')
    - **ORO** — **GENERAL** — guzti ('todos')
- **ESP** — **LOCUCIÓN** — noizean behin ('de vez en cuando')
- **ITJ** — **INTERJECCIÓN** — alajaina ('caramba')
- **IZE** — **NOMBRE**
  - **ARR** — **COMÚN** — zuhaitz

IZB	**PROPIO (DE PERSONA)**	Mikel
LIB	**PROPIO (DE LUGAR)**	Donostia
ZKI	**NÚMERO**	bat (‘uno’)

- **IOR** — **PRONOMBRE**
  - *PER* — *PERSONAL*
    - **PERARR** — **COMÚN** — ni
    - **PERIND** — **INTENSIFICADO** — neu ('yo mismo')
  - *IZG* — *INDEFINIDO*
    - **IZGMGB** — **COMÚN** — norbait
    - **IZGGAL** — **INTERROGATIVO** — nor
  - **BIH** — **REFLEXIVO** ('se') — -(r)en burua
  - **ELK** — **RECÍPROCO** — elkar ('se')
- **LOT** — **CONECTOR**
  - **LOK** — **CONECTOR ORACIONAL** — hala ere ('sin embargo')
  - **JNT** — **CONJUNCIÓN** — edo ('o')
- **PRT** — **PARTÍCULA**

Categorías auxiliares (5)

- **ADL** — **VERBO AUXILIAR** — du ('ha ...')
- **ADT** — **VERBO SINTÉTICO** ('viene' (ahora))
- **SIG** — **SIGLA** — EHU ('UPV')
- **SNB** — **SÍMBOLO** — a
- **LAB** — **ABREVIATURA** — etab. ('etc.')

7.2. Categorías morfológicas (8)

- **AMM** — **MORFEMA DE CLASE VERBAL** — -tu (participio)
- **ASP** — **MORFEMA ASPECTUAL** — -tuko (futuro)
- **ATZ** — **SUFIJO** (*harkaizpe* ('lugar bajo la roca')) — -pe
- **AUR** — **PREFIJO** — ber-(*berregin*

('rehacer'))

- **DEK** **ARTÍCULO + MORFEMA DE DECLINACIÓN** -aren ('del')
- **ELI** **ELIPSIS** ø (*etxearenØa*)
- **ERL** **CONJUNCIÓN SUBORDINANTE** -(e)la ('que')
- **GRA** **MORFEMA DE COMPARACIÓN** -ago ('más...')
- **MAR** **GUIÓN** – (zabor-garbitzaile ('basurero'))

7.3. Signos de puntuación

- **PNT** **PUNTO**
- **BSP** **OTROS SIGNOS DE PUNTUACIÓN** (que se asemejan al punto: dos puntos, punto y coma...)
- **BER** **OTROS SÍMBOLOS DE PUNTUACIÓN** (paréntesis, comas, signo de nterrogación…)

8. Anexo 2: Formato y contenido del diccionario bilingüe

A continuación se presentan la estructura y una pequeña parte del diccionario bilingüe. El formato está exhaustivamente descrito en el capítulo 2 de la referencia [1].

El diccionario bilingüe reside en *matxin/src/eseu.xml*

8.1. Estructura

Las entradas del diccionario están juntas, pero organizadas en diferentes secciones (*sections*):

- diccionario principal
- categorías cerradas
- casos especiales
- preposiciones

8.2. Contenido

8.2.1. Diccionario principal

Primeramente se analizan los términos en es que aparecen en el diccionario principal,

El diccionario principal (*main*) tiene este aspecto:

```
<e><p><l>adherir</l><r>itsatsi</r></p><par n='VM_ADI_SIN'/></e>
<e><p><l>adhesión</l><r>itsaspen</r></p><par n='NC_IZE_ARR'/></e>
<e><p><l>adhesivo</l><r>itsaskor</r></p><par n='AQ_ADJ_IZO'/></e>
```

Donde las etiquetas expresan lo siguiente:

<e>: expresa una entrada

<p>: abarca los pares español-euskera

<l>: expresa el elemento de la izquierda, que, en este caso es el español

<r>: expresa el elemento de la derecha, que, en este caso es el euskera

<par>: hace referencia a un paradigma. En el diccionario general se toma como referencia en la transferencia el lema y no la forma. En este paradigma aparecen la categoría de ese lema en español, y la categoría y subcategoría de ese lema en euskera, para conseguir transformar correctamente la información morfológica.

Así el término 'adherir' tiene la etiqueta 'verbo' (VM) en *freeling*, y su par en euskera ('itsatsi') también tiene la etiqueta de 'verbo' (ADI_SIN). Estas etiquetas se expresan en el paradigma para realizar la transformación de etiquetas correctamente.

El término 'adhesión' tiene la etiqueta 'nombre' (NC) en *freeling*, y su par en euskera ('itsaspen') también tiene la etiqueta de 'nombre' (IZE_ARR).

El término 'adhesivo' tiene la etiqueta 'adjetivo' (AQ) en *freeling*, y su par en euskera ('itsaskor') también tiene la etiqueta de 'adjetivo' (ADJ_IZO).

A veces, la categoría del español y la del euskera no son la misma:

```
<e><p><l>antiviral</l><r>birus kontrako</r></p><par n='NC_ADJ_IZL'/></e>
```

En este caso 'antiviral' aparece como 'nombre' en freeling (NC) y como 'adjetivo' (ADJ_IZL) en euskera.

El significado de las etiquetas morfosintácticas del euskera se ha explicado en el apéndice 1.

8.2.2. Paradigmas en el diccionario bilingüe

Aunque la función primitiva de los paradigmas es controlar el análisis y la generación morfológica, también se ha usado para la transformación de información morfológica en el diccionario bilingüe.

He aquí dos ejemplos de las definiciones de estos paradigmas que nos van a servir para explicar el funcionamiento.

En un primer ejemplo podemos ver el caso donde no se produce ninguna transformación, ya que en la parte *l* aparece la etiqueta en *es* y en la parte r las etiquetas en *eu*. Aquí el uso del paradigma vale para que en euskera se divida la etiqueta en dos partes.

```
<pardef n='VM_ADI_SIN'>
    <e><p> <l><s n='VMG0000'/></l>
           <r><s n='ADI'/><s n='SIN'/></r>  </p></e>
    <e><p> <l><s n='VMIC1P0'/></l>
           <r><s n='ADI'/><s n='SIN'/></r>  </p></e>
    <e><p> <l><s n='VMIC1S0'/></l>
           <r><s n='ADI'/><s n='SIN'/></r>  </p></e>
    <e><p> <l><s n='VMIC2P0'/></l>
           <r><s n='ADI'/><s n='SIN'/></r>  </p></e>
    <e><p> <l><s n='VMIC2S0'/></l>
           <r><s n='ADI'/><s n='SIN'/></r>  </p></e>
    <e><p> <l><s n='VMIC3P0'/></l>
           <r><s n='ADI'/><s n='SIN'/></r>  </p></e>
    ...
    ...
</pardef>
```

En el segundo caso, además de separar las etiquetas en euskera, se obtiene información adicional de la etiqueta en castellano, a menudo + para indicar que se añadirán sufijos (<s n='+'/>)

```
<pardef n='NC_IZE_ARR'>
    <e><p> <l><s n='NCCN000'/></l>
           <r><s n='IZE'/><s n='ARR'/></r>  </p></e>
    <e><p> <l><s n='NCCP000'/></l>
           <r><s n='IZE'/><s n='ARR'/><s n='+'/><s n='NUMP'/></r>     </p></e>
    <e><p> <l><s n='NCCS000'/></l>
           <r><s n='IZE'/><s n='ARR'/><s n='+'/><s n='NUMS'/></r>     </p></e>
    <e><p> <l><s n='NCFN000'/></l>
           <r><s n='IZE'/><s n='ARR'/></r>        </p></e>
    <e><p  ><l><s n='NCFP000'/></l>
           <r><s n='IZE'/><s n='ARR'/><s n='+'/><s n='NUMP'/></r>     </p></e>
    <e><p> <l><s n='NCFS000'/></l>
           <r><s n='IZE'/><s n='ARR'/><s n='+'/><s n='NUMS'/></r>     </p></e>
    <e><p> <l><s n='NCMN000'/></l>
           <r><s n='IZE'/><s n='ARR'/></r>        </p></e>
    <e><p> <l><s n='NCMP000'/></l>
           <r><s n='IZE'/><s n='ARR'/><s n='+'/><s n='NUMP'/></r>     </p></e>
```

```
      <e><p> <l><s n='NCMS000'/></l>
             <r><s n='IZE'/><s n='ARR'/><s n='+'/><s n='NUMS'/></r>     </p></e>
   </pardef>
```

8.2.3. Categorías cerradas

En el diccionario de las categorías cerradas, tenemos los lemas de *es* con su correspondiente etiqueta de *freeling*. En la etiqueta de *freeling*, además de aparecer la marca de la categoría a la que pertenece (p. ej. DD: determinante), aparece la información morfológica de la forma que aparece en el texto (MS, FS, FP, MP).

```
<e><p> <l>aquel<s n='DD0MS0'/></l>
       <r>hura<s n='DET'/><s n='ERKARR'/><s n='+'/><s n='NUMS'/><s n='+'/></r> </p></e>
<e><p> <l>aquel<s n='DD0FS0'/></l>
       <r>hura<s n='DET'/><s n='ERKARR'/><s n='+'/><s n='NUMS'/><s n='+'/></r> </p></e>
<e><p> <l>aquel<s n='DD0FP0'/></l>
       <r>haiek<s n='DET'/><s n='ERKARR'/><s n='+'/><s n='NUMP'/><s n='+'/></r> </p></e>
<e><p> <l>aquel<s n='DD0MP0'/></l>
       <r>haiek<s n='DET'/><s n='ERKARR'/><s n='+'/><s n='NUMP'/><s n='+'/></r> </p></e>
```

En el lado del euskera (<r>) además de aparecer el lema y su correspondiente categoría y subcategoría, tenemos información morfológica para poder hacer la trasferencia adecuada. Si para hacer la trasferencia no hay ningún lema (y por supuesto ni categoría ni subcategoría) se usa un guión como marca: "<r>-"

Posteriormente al campo, (detrás de <s n='+'/>) aparece la información morfológica que debe llevar el sintagma.

8.2.4. Casos especiales

En los casos en que ponemos que en euskera no tenemos un lema adecuado para hacer esa transferencia, es debido a que hemos de hacer esa trasferencia utilizando una posposición. Este es el caso de la trasferencia de "alguno".

```
<e><p> <l>aquel<s n='DD0MS0'/></l>
       <r>hura<s n='DET'/><s n='ERKARR'/><s n='+'/><s n='NUMS'/><s n='+'/></r> </p></e>

<e><p><l>alguno<s n='DI0FP0'/></l>
       <r>-<s n='+'/><s n='MG'/><s n='+'/><s n='[DEK][GEN]++batzuk'/></r>      </p></e>

<e><p> <l>mi<s n='DP1CSS'/></l>
       <r>nire<s n='ADJ'/><s n='IZL'/><s n='+'/><s n='NUMS'/><s n='+'/></r>    </p></e>
```

En otros casos, en la parte del euskera, no aparece ningún lema ni categoría, y lo que se utiliza para la traducción es una posposición: por ejemplo '-en batzuk'

```
<e><p> <l>alguno<s n='DI0FP0'/></l>
       <r>-<s n='+'/><s n='MG'/><s n='+'/><s n='[DEK][GEN]++batzuk'/></r>      </p></e>
```

Tras el campo que marca la morfología, aparece otro separador de campo, y después la información que hemos de pasar al sintagma (que es la correspondiente a "alguno"): <s n='[DEK][GEN]++batzuk'/>

Primero se traducen al euskera todos los términos, y luego, en la generación, se consulta otro diccionario (de generación) para crear los términos que necesitan de información morfológica

más profunda, ya que el lema no siempre es verdaderamente un lema en euskera y se necesita su análisis morfológico a la hora de la generación

```
<e><p> <l>mi<s n='DP1CSS'/></l>
       <r>nire<s n='ADJ'/><s n='IZL'/><s n='+'/><s n='NUMS'/><s n='+'/></r>    </p></e>
```

8.2.5. Preposiciones

En las preposiciones no tenemos información morfológica del sintagma ni del lema (debido que en euskera aparecen posposiciones). Por ello, tras el guión en el campo correspondiente al lema, categoría y subcategoría, aparecen dos marcadores de "cambio de campo" y en último lugar aparece la información del lema, que es la que hemos de pasar al sintagma.

```
<e><p> <l>con<s n='SPS00'/></l>
       <r><s n='+'/><s n='+'/><s n='+'/><s n='+'/>con<s n='+'/></r>      </p></e>
<e><p> <l>contra<s n='SPS00'/></l>
       <r><s n='+'/><s n='+'/><s n='+'/><s n='+'/>contra<s n='+'/></r> </p></e>
```

8.2.6. Términos multipalabra

Los términos multipalabra que pueden pertenecer tanto a las categorías cerradas, como a las preposiciones, se tratan igual que el resto de palabras.

Hay que reseñar que el espacio entre las palabras se codifica con el signo “_”, pues así es como freeling da el análisis de esas palabras.

Por ejemplo:

```
<e>    <p>    <l>importar_un_pimiento</l>
              <r>bost_axola_izan</r>    </p>
       <par n='VM_ADI_SIN'/><p><l/><r><s n='+'/><s n='+'/><s n='+'/></r></p>   </e>
<e>    <p>    <l>importar_un_pito</l>
              <r>bost_axola_izan</r>    </p>
       <par n='VM_ADI_SIN'/><p><l/><r><s n='+'/><s n='+'/><s n='+'/></r></p>   </e>
<e>    <p>    <l>importar_un_rábano</l>
              <r>bost_axola_izan</r>    </p>
       <par n='VM_ADI_SIN'/><p><l/><r><s n='+'/><s n='+'/><s n='+'/></r></p>   </e>
<e>    <p>    <l>imprudencia_temeraria</l>
              <r>zuhurtziagabetasun_ausartegi</r>    </p>
       <par n='NC_IZE_ARR'/><p><l/><r><s n='+'/><s n='+'/><s n='+'/></r></p>   </e>
<e>    <p>    <l>impuesto_sobre_actividades_económicas</l>
              <r>jarduera_ekonomikoen_gaineko_zerga</r>     </p>
       <par n='NC_IZE_ARR'/><p><l/><r><s n='+'/><s n='+'/><s n='+'/></r></p>   </e>
<e>    <p>    <l>impuesto_sobre_la_renta</l>
              <r>errentaren_gaineko_zerga</r>  </p>
       <par n='NC_IZE_ARR'/><p><l/><r><s n='+'/><s n='+'/><s n='+'/></r><      /p></e>
<e>    <p>    <l>delante_de<s n='SPS00'/></l>
              <r><s n='+'/><s n='+'/><s n='+'/><s n='+'/>delante_de<s n='+'/></r></p></e>
<e>    <p>    <l>debajo_de<s n='SPS00'/></l>
              <r><s n='+'/><s n='+'/><s n='+'/><s n='+'/>debajo_de<s n='+'/></r></p></e>
```

9. Anexo 3: Contenido del diccionario morfológico

A continuación se presentan la estructura, formato y una pequeña parte del diccionario morfológico.

9.1. Estructura

Se diferencian varias secciones

- alfabeto (*alphabet*): especifica los caracteres que se usan en el diccionario
- definición de atributos (*sdefs*): especifica los atributos de información morfológica que se utilizan
- definición de paradigmas (*pardefs*): se especifican los paradigmas de declinación (más de 300), apareciendo por cada entrada de un paradigma, por un lado (l) la cadena de caracteres (pseudomorfema) que se añade a la raíz (pseudolema), y por el otro la información morfológica correspondiente.
- sección principal (*main*): aparecen las raíces con los paradigmas correspondientes.

Para resolver el tema de los prefijos, estos se han incluido en la sección principal y dentro las entradas de su paradigma hay raíces que tiene asignados nuevos paradigmas.

9.2. Contenido

Cabecera y alfabeto

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE dictionary SYSTEM "http://www.torsimany.ua.es/esgleuca/dix.dtd">
<dictionary>
  <alphabet>·ÀÁÂÄÇÈÉÊËÌÍÎÏÑÒÓÔÖÙÚÛÜàáâäçèéêëìíîïñòóôöùúûüABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrstuvwxyz</alphabet>
```

Atributos

```
  <sdefs>
    <sdef n="NR_HK"/>
    <sdef n="BST"/>
    <sdef n="ZIU"/>
    <sdef n="MDNC"/>
    <sdef n="B5A"/>
    <sdef n="LOK"/>
    <sdef n="DZG"/>
    <sdef n="B5B"/>
    <sdef n="NI_HI"/>
    <sdef n="ERG"/>
     ...
    <sdef n="ITJ"/>
    <sdef n="NK_NI"/>
  </sdefs>
```

Paradigmas

```
  <pardefs>
    <pardef n="A11_37">
```

```
      <e>
        <p>
          <l>e</l>
          <r><j/><s n="AMM"/><s n="ADOIN"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>e</l>
          <r><j/><s n="AMM"/><s n="PART"/><j/><s n="ASP"/><s n="BURU"/></r>
        </p>
      </e>
      <e>
        <p>
         <l>earaz</l>
          <r>arazi<s n="ATZ"/><s n="ADI"/><s n="SIN"/><j/><s n="AMM"/><s n="ADOI
N"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>earazi</l>
          <r>arazi<s n="ATZ"/><s n="ADI"/><s n="SIN"/><j/><s n="AMM"/><s n="PART
"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>earazi</l>
          <r>arazi<s n="ATZ"/><s n="ADI"/><s n="SIN"/><j/><s n="AMM"/><s n="PART
"/><j/><s n="ASP"/><s n="BURU"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>earazi</l>
          <r>arazi<s n="ATZ"/><s n="ADI"/><s n="SIN"/><j/><s n="AMM"/><s n="PART
"/><j/><s n="DEK"/><s n="ABS"/><s n="MG"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>earazte</l>
          <r>arazi<s n="ATZ"/><s n="ADI"/><s n="SIN"/>te<s n="ATZ"/><s n="IZE"/>
<s n="ARR"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>earazte</l>
          <r>arazi<s n="ATZ"/><s n="ADI"/><s n="SIN"/>te<s n="ATZ"/><s n="IZE"/>
<s n="ARR"/><j/><s n="DEK"/><s n="ABS"/><s n="MG"/></r>
        </p>
      </e>
      ...
    </pardef>
    <pardef n="I0">
      <e>
        <p>
          <l>0</l>
          <r></r>
        </p>
      </e>
    </pardef>
```

```
    <pardef n="I11_28">
      <e>
        <p>
          <l>r</l>
          <r></r>
        </p>
      </e>
      <e>
        <p>
          <l>rdaino</l>
          <r><j/><s n="DEK"/><s n="ABU"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>rdainoko</l>
          <r><j/><s n="DEK"/><s n="ABU"/><j/><s n="DEK"/><s n="GEL"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>rdainoko</l>
          <r><j/><s n="DEK"/><s n="ABU"/><j/><s n="DEK"/><s n="GEL"/><j/><s n="D
EK"/><s n="ABS"/><s n="MG"/></r>
        </p>
      </e>
      <e>
        <p>
          <l>rdanik</l>
          <r><j/><s n="DEK"/><s n="ABL"/></r>
        </p>
      </e>
      ...
    </pardef>
```

Lemas tras sufijos

```
    <pardef n="ADITZAK1">
    <e>
        <p>
          <l>abes</l>
          <r><s n="ADI"/><s n="SIN"/></r>
      </p>
      <par n="A14_41"/>
    </e>
    <e>
        <p>
          <l>alderanz</l>
          <r><s n="ADI"/><s n="SIN"/></r>
      </p>
      <par n="A14_41"/>
    </e>
    ...
    </pardef>
  </pardefs>
```

Sección principal y sufijos

```
  <section id="main" type="standard">
    <e>
      <p>
        <l>ba</l>
```

```
      <r>ba<j/><s n="ERL"/><s n="MEN"/><s n="BALD"/></r>
    </p>
    <par n="BABALD"/>
  </e>
  <e>
    <p>
      <l>ba</l>
      <r>ba<j/><s n="PRT"/><s n="EGI"/></r>
    </p>
    <par n="BABAI"/>
  </e>
  <e>
    <p>
      <l>bait</l>
      <r>bait<j/><s n="ERL"/><s n="MEN"/><s n="KAUS"/></r>
    </p>
    <par n="BAIT1"/>
  </e>
  <e>
    <p>
      <l>bai</l>
      <r>bait<j/><s n="ERL"/><s n="MEN"/><s n="KAUS"/></r>
    </p>
    <par n="BAIT2"/>
  </e>
  <e>
    <p>
      <l>berr</l>
      <r>ber<j/><s n="AUR"/></r>
    </p>
    <par n="ADITZAK1"/>
  </e>
  <e>
    <p>
      <l>bir</l>
      <r>ber<j/><s n="AUR"/></r>
    </p>
    <par n="ADITZAK2"/>
  </e>
```

Lemas comunes

```
  <e>
    <p>
      <l>koip</l>
      <r><s n="ADI"/><s n="SIN"/></r>
    </p>
    <par n="A11_37"/>
  </e>
  <e>
    <p>
      <l>kost</l>
      <r><s n="ADI"/><s n="SIN"/></r>
    </p>
    <par n="A11_39"/>
  </e>
  <e>
    <p>
      <l>pas</l>
      <r><s n="ADI"/><s n="SIN"/></r>
    </p>
    <par n="A11_39"/>
  </e>
  ...
```

```
    <e>
      <p>
        <l>zernahi</l>
        <r><s n="DET"/><s n="NOLARR"/><s n="MG"/></r>
      </p>
      <par n="ZB2_40"/>
    </e>
  </section>
</dictionary>
```