Un paso adelante

Table of Contents

Un paso adelante

Plan de tecnologías lingüísticas libres

Juan Rafael Fernández García

Coordinador responsable de herramientas lingüísticas TLDP−ES

Copyright © 2003 por Juan Rafael Fernández García

$Id: un−paso−adelante.xml,v 1.3 2003/09/08 21:42:57 juanfernandez Exp $

Historial de revisiones

Revisión 0.7	2003−09−07	Revisado por: jrf
Versión enviada al VI Congreso Hispalinux		
Revisión 0.6	2003−03−17	Revisado por: jrf
Superada tortura de xmllint, victorioso		
Revisión 0.5	2003−02−26	Revisado por: jrf
La información pasa a doc−traduccion−libre		
Revisión 0.4	2003−02−25	Revisado por: jrf
Modificaciones menores del marcado		
Revisión 0.3	2003−02−24	Revisado por: jrf
Revisión del xml		
Revisión 0.2	2003−02−20	Revisado por: io
Primera versión xml		
Revisión 0.1	2002−17−12	Revisado por: jrf
Versión inicial (txt)		

Notas previas para una especificación de requisitos de las herramientas lingüísticas de TLDP (La comprensión plena de este documento requiere familiaridad con las herramientas lingüísticas o la lectura previa de doc−traducción−libre http://es.tldp.org/Articulos/0000otras/doc−traduccion−libre/ , documento producido en el seno de *TLDP−ES)*

En este artículo el autor presenta y justifica sus propuestas con respecto a la adopción de estándares libres de marcado e intercambio de información lingüística y el desarrollo de herramientas libres de ayuda a la escritura y traducción, de manera que puedan someterse a discusión pública y adoptarse como especificaciones de TLDP. Es en cierto modo un apéndice ejecutivo, las conclusiones de la exposición realizada en el documento anterior.

- Última versión: http://es.tldp.org/especificaciones/herramientas−linguisticas/herramientas−linguisticas/
- Fuente: http://cvs.hispalinux.es/cgi−bin/cvsweb/esp−herramientas−linguisticas/un−paso−adelante.xml

---

Tabla de contenidos

1. Introducción

La temática de este documento es lo que en la jerga de la industria se llama *localization* (no me atrevo yo a traducir «localización»), directamente relacionada con la traducción y sus herramientas. Me he esforzado en atenerme a desarrollos posibles a corto y medio plazo, pero continuamente me ha asaltado la tentación de entrar en el tema de la traducción automática. Queremos, a corto plazo, que las máquinas nos ayuden a traducir; a medio plazo el objetivo es que traduzcan por nosotros. Lo que pasa es que los dos objetivos convergen en dotar de inteligencia a las máquinas. Pienso que cualquier asistente a la traducción necesita reglas: morfológicas, sintácticas, semánticas, necesita una gramática y un «conocimiento del mundo». ¿Cómo traduciría un programa sin inteligencia oraciones como spirits sold here? ¿cómo puede un humilde corrector ortográfico distinguir a de ha?

Es la hora de que el software libre llegue a la madurez. Eso implica también que conozcamos las tendencias de la industria y adoptemos y trabajemos con los estándares. Este documento trata de señalar el camino para el desarrollo de nuestras herramientas de CAT, con el ojo puesto en el desarrollo futuro de herramientas de traducción más o menos automática. Ahora hay que ponerse técnicos y me temo que la lectura no va a ser fácil; no soy un experto y escribo mientras aprendo así que es probable que se encuentren inexactitudes que será conveniente corregir. De todos modos no pretendo más de plantear unas perspectivas y unas posibles líneas de desarrollo; está claro que LuCAS y TLDP son ejemplos de éxito como modelos de cooperación libre y que el actual proceso de creación de la editorial libre y la conversión de la documentación a DocBook XML son grandes ideas.

Dice Ismael Olea, hablando del desarrollo de TLDP,

> Hay dos niveles de objetivos a cumplir:
>
> - herramientas para el fácil mantenimiento y traducción de nuestra documentación
> - herramientas lingüísticas de calidad

Empezar a trabajar en la segunda opción no debe suponer merma de la primera.

1.1. ¿Por qué?

Pensando sobre el esquema de Ismael Olea Arquitectura para reproducción y publicación del borrador de *LuCAS v4: creando el sistema de publicación de TLDP−ES* (http://cvs.hispalinux.es/cgi−bin/cvsweb/doc−conf−creando−lucasv4/herramientas−reproduccion−publicacion.png), creo que es el momento intentaré explicar por qué de integrar en el esquema las herramientas lingüísticas y dar el paso de adoptar los estándares y potencialidades de la ingeniería lingüística actual.

Dependiendo de cómo nos planteemos la realidad de nuestro fondo documental, podemos pensar en él como en una gran biblioteca electrónica, como un repositorio digital (como el DSpace del MIT) o como un gran corpus lingüístico. Cada punto de vista plantea posibilidades distintas: en tanto que biblioteca, debemos escuchar las aportaciones de los bibliotecónomos. Debemos también acordar los metadata y el formato de registro de los documentos, para que las búsquedas sean cada vez más eficaces. Yo, como lingüista, propongo que lo miremos con un nuevo enfoque: somos un gran equipo de traductores y redactores de documentación. Una de nuestras prioridades debe ser mejorar nuestras herramientas de ayuda a la escritura.

Disponemos de un fondo de varios cientos de megas de documentación técnica, en proceso de transcripción a DocBook XML, en su mayor parte traducciones de obras o documentación de tema informático que también son de licencia libre y disponemos además de las traducciones a multitud de idiomas. Ninguna empresa ¿quizás alguna universidad? dispone de un fondo documental como el nuestro para la elaboración de terminologías específicas, para la confección de un corpus tan completo de la materia ni para la creación de memorias de traducción. Ninguna empresa dispone del número de traductores de que dispone la comunidad GNU Linux. Y sin embargo las características de esta comunidad (en su mayor parte estudiantes y profesionales de la informática) hace que el modo de trabajar y las herramientas estén a años luz de las posibilidades que presenta la investigación actual. No se conocen las experiencias que se realizan en la Unión Europea ni las herramientas que se utilizan, ignoramos los proyectos universitarios y los intentos de estandarización que está realizando la industria.

Y la distancia con respecto a la evolución del software propietario, lamentablemente, se amplía en lugar de acortarse. La comunidad, bajo el nombre de HispaLinux, o de TLDP o de FSF o del que sea, debe funcionar como una entidad y personarse y participar en los foros donde se están creando estándares (LISA, ISLE) y en los proyectos financiados por la Unión Europea (debemos aspirar a recibir financiación de las instituciones y de las empresas). Es la hora de preguntarse si el modelo de voluntarios a tiempo parcial es eficaz para los objetivos que nos proponemos.

1.1.1. La crisis: más allá de los ficheros .po

No podemos olvidar que gettext y .po resuelven algunos de los problemas con los que tiene que enfrentarse todo software de traducción: la extracción de las cadenas que deben ser traducidas, la segmentación y la referencia al lugar en el código al que pertenece la cadena, la alineación entre texto fuente y texto traducido, la capacidad de reutilizar traducciones cuando el fichero original ha cambiado, marcando las cadenas equivalentes y aquellas en las que la traducción es sólo aproximada (*fuzzy*) o no existe, y el informe automático del número de cadenas traducidas

Existen varios intentos de ampliar el uso de nuestras herramientas a textos libres (ficheros no .po, xml sobre todo), mediante el procedimiento de convertir de alguna manera los documentos a ficheros .po:

- *doc−i18n−tool* (http://mail.gnome.org/archives/gnome−doc−list/2001−October/msg00034.html)
- *poxml* (de KDE)
- *po−debiandoc*
-

po−pod,nuevo miembro de la familia de utilidades po−for−everything (po4a), tal como se presenta en la Debian Weekly News del 3 de dic. de 2002:

" The goal of po−pod is to allow translators to work only with well known po files when translating pod documentation. The goal of po4a is to ease translations (and more interestingly, the maintainance of translations) by using gettext tools on areas where they were not yet expected. " (http://lists.debian.org/debian−i18n−0211/msg00009.html)

La referencia es http://www.ens−lyon.fr/~mquinson/deb.html#po−pod.

Llevo tiempo preguntándome si estos tímidos intentos van bien encaminados. Mis objeciones son las siguientes:

- .po no es un formato bien marcado (me refiero, cualquiera que haya luchado con el Robot del Proyecto de Traducción Libre me entenderá, a que quede claro que en la tercera línea debe ir el *e−mail* del traductor y el *copyright*, por ejemplo)

- Me temo que .po no permite granularidad en la descomposición del párrafo/cadena en sus componentes. Cuando reconoce una cadena como *fuzzy* está actuando únicamente al nivel de *grep*.

Aunque a efectos del resultado sería indiferente el formato que utilizara internamente una utilidad de traducción, en nuestro análisis debemos considerar si el formato elegido es adecuado para nuestras necesidades (ya no la traducción de cadenas en tanto que mensajes de un programa informático, sino de documentación).

Quizás el camino no esté en convertir los ficheros xml en ficheros .po para poder traducirlos, sino por el contrario mantener gettext y el proceso de extracción de mensajes, que funciona bien y es estable, convertirlos en ficheros xml bien formados, y aplicar herramientas generalizadas que puedan utilizarse sobre cualquier fichero xml. Lo que propondría sería una especie de .po con marcas, utilizable además para texto libre.

---

2. Al grano

En las secciones que restan de este documento vamos a seguir el procedimiento de discutir brevemente las necesidades y herramientas libres conocidas y acabaremos por hacer una propuesta, bien de adopción de un estándar o desarrollo de una aplicación, bien de estudio previo a tomar una decisión.

Son de varios géneros las utilidades de ayuda a la escritura y al traductor: diccionarios, concordancias, memorias de traducción, analizadores de todas clases. No hay espacio en este artículo para volver a hacer lo que como decíamos en el resumen inicial ya está hecho: subyace a todas las referencias la familiaridad tácita con las herramientas o con http://es.tldp.org/Articulos/0000otras/doc−traduccion−libre/ . Es el momento de una revisión detallada y de tomar decisiones.

2.1. Propuesta: Líneas generales

En TLDP se usarán estándares libres y abiertos: DocBook, TEI, XCES y cuando se utilice uno propio se crearán mecanismos para la importación/conversión (recordemos por ejemplo la utilidad incorporada a Mimers brunn como conversor de .po a TMX).

El formato de los documentos será preferentemente DocBook XML, porque permite la separación lógica entre los niveles semántico y de presentación y da pie a desarrollos relacionados con la web semántica, y las herramientas trabajarán sobre XML.

La codificación UTF−8 es lógica en un proyecto con ambiciones de universalidad.

2.2. Diccionarios

En primer lugar evitemos una confusión generalizada, *de verba non est disputandum*. Diccionarios son listas de palabras de un idioma, entradas léxicas con su definición, tablas de equivalencias entre dos o más lenguas y cualesquiera otras variaciones que se nos planteen; nomenclatura, glosario, lexicón, lemario etc. son términos que se usan de manera no estable en la literatura y no deben impedir que nos entendamos[1]. Mucho más interesante es plantearnos el campo de realidad cubierto por el diccionario y la información que proporciona (morfológica, de uso, terminológica). Por último, distinguiremos diccionarios destinados a ser usados por humanos (en forma de libro o como consultas a través de una interfaz) de aquellos creados para ser usados por máquinas. También deberemos distinguir los que tienen una ambición descriptiva y verbal de los diccionarios terminológicos, que son como sabemos normativos y su objeto son los conceptos de un campo. Al fin y al cabo por lo pronto sólo queremos traducir documentación informática ¿o no?

2.2.1. Discusión

Estamos en LuCAS−desarrollo reimpulsando nuestro propio diccionario inglés−español de informática Giait[2] Por otro lado estamos intentando crear el mayor lemario del español[3]. Y periódicamente nos planteamos qué sentido tendría reclamar la liberación del diccionario de la RAE (probablemente muy escaso). Pero todos estos son pequeños pasos. Ponernos a escribir diccionarios nos convierte en lexicógrafos; pero escribir un diccionario técnico nos obliga a plantearnos los problemas de la terminología. Además, si logramos crear una estructura para la confección de terminologías (partiendo de nuestro corpus de informática pero no quedándonos ahí) podremos crear glosarios y diccionarios, presentar colocaciones y listas de ejemplos. Y hacer un uso normativo (estandarizado, coherente) de nuestra terminología. Estaremos por ejemplo en condiciones de definir como término informático el que aparece en nuestro fondo documental y no aparece recogido en nuestro diccionario general de español (por ahora inexistente).

Pero vayamos poco a poco: un diccionario pensado para su impresión en papel o en pantalla, destinado a ser consultado por humanos, no es lo mismo que una base de datos terminológicos (un lexicón computacional), pensada para su utilización en la traducción automática.

2.2.1.1. Sobre TEI P4

La versión XML de las *TEI P4 Guidelines*[4], en su capítulo 13 Terminological Databases, advierte

> Since its first publication, this chapter has been rendered obsolete in several respects, chiefly as a result of the publication of ISO 12200, and a variant of it (TBX) which has been recently adopted by LISA. Work is currently ongoing in the ISO community to define a generic platform for terminological markup (ISO CD 16642, TMF: Terminological Markup Framework), in the light of which it is anticipated that the recommendations of the present chapter will be substantially revised.

Sí nos interesa el capítulo 12 Print Dictionaries.

> A simple dictionary entry may contain information about the form of the word treated, its grammatical characterization, its definition, synonyms, or translation equivalents, its etymology, cross−references to other entries, usage information, and examples.

Interesante parece estudiar como ejemplo de aplicación de TEI
http://www.human.toyogakuen−u.ac.jp/~acmuller/articles/ddb−ebti2001.htm

---

2.2.1.2. Sobre WordNet

Princeton WordNet (http://www.cogsci.princeton.edu/~wn/w3wn.html)

`man wndb` ó

http://www.cogsci.princeton.edu/~wn/man1.7.1/wndb.5WN.html

EuroWordNet (http://www.illc.uva.nl/EuroWordNet/). EuroWordNet was a European resources and development project (LE−2 4003 & LE−4 8328) supported by the Human Language Technology sector of the Telematics Applications Programme, 1996−1999

Problema

Me temo que la versión europea de wn tiene carácter no libre y sus herramientas son no libres (editor Polaris; el visor Periscope solamente es freeware)

2.2.2. Propuesta

1. El formato estándar para la distribución de diccionarios es .dict (un éxito, multiplicado casi por cuatro desde la primera redacción de doc−traduccion−libre)

2. Para consultas el estándar es la red dict, mediante clientes específicos o interfaz *web*

3. Otro tema es cómo escribir diccionarios: propongo con dudas adoptar TEI P4 como la DTD de nuestros diccionarios específicos. En este sentido, es necesario también completar el desarrollo de Giait y desarrollar herramientas generales que puedan utilizarse para el desarrollo de nuevos diccionarios

4. La gran pregunta, que normalmente no se hace cuando se está haciendo un diccionario, es qué información debe contener. Si queremos poder utilizar herramientas avanzadas de ayuda a la traducción nuestro diccionario debe tener información gramatical leíble por máquinas. Propongo el uso de XCES.

5. Cómo incorporar términos a los diccionarios: ¿modelo ORCA? (contribuciones públicas, revisadas por un moderador) ¿modelo lista spanglish? (discusiones inter pares)

   Utilizar herramientas de confección de glosarios. Debemos ir hacia diccionarios basados en *corpus*, con herramientas de extracción terminológica que cubren de forma exhaustiva un campo (y nos señalan fehacientemente qué términos faltan por definir) y abandonar el método manual de adición de entradas.

6. Cómo garantizar la calidad de las incorporaciones: Crear sistema de mantenimiento de calidad (incluir entre las marcas de cada término autor, fecha, revisión)

Será necesario adaptar TEItools (http://xtalk.msk.su/SGML/TEItools) son para TEI Lite, no para TEI P4.

2.2.3. Un caso práctico

Las dos primeras entradas de la letra «a» en las fuentes del diccionario Giait son

```
@@ING A, Ampere, Amps
@@CAS Ampère (amperio), amperios.
@@DIC Vocablo definido únicamente para el Diccionario
@@FIN

@@ING A Programming Language", APL
@@CAS Lenguaje de Programación APL
```

```
@@GLO APL es la sigla de "A Programming Language" (Un Lenguaje de
@@GLO Programación), que fue un libro escrito en 1962 por el credor del
@@GLO lenguaje, Kenneth E. Iverson. Basado en lo que antes se había conocido
@@GLO como la Nomenclatura Iverson, el APL es un lenguaje de programación
@@GLO extremadamente conciso, diseñado para el manejo de los arreglos
@@GLO (arrays). Los arreglos pueden ser escalares, vectoriales, tablas o
@@GLO matrices de dos o más dimensiones, pudiendo estar compuestos de
@@GLO información numérica o alfanumérica. Bajo la conducción de Iverson,
@@GLO IBM introdujo en 1966 el APL\360.
@@GLO Como el APL evita la introducción de las computadoras personales,
@@GLO originalmente se lo empleó unicamente en mainframes. Desde 1983, han
@@GLO estado disponibles las versiones de APL para las PC. Debido a su
@@GLO conjunto de caracteres especiales expandidos, el APL requiere un
@@GLO teclado especial, o el uso de macros, para el ingreso de datos.
@@GLO Las versiones actuales de APL para mainframes y PC se denominan
@@GLO APL2.
@@GLO
@@FIN
```

Convertidas a formato .dict (descomprimido) actualmente quedan así:

A, Ampere, Amps Ampère (amperio), amperios.

A Programming Language", APL Lenguaje de Programación APL

Vemos que, aparte de errores tipográficos que habrá que corregir, se ha perdido la información de glosario y el marcado (lo que es inglés, castellano, comentario, glosario)

¿Cómo aparece?

```
11262928 23 n 03 ampere 0 amp 0 A 0 003 @ 11259168 n
0000 #p 11263273 n 0000 %p 11263164 n 0000 | the basic unit of
electric current adopted under the System International d'Unites;
"a typical household circuit carries 15 to 50 amps"
```

¿Cómo quedaría todo esto en TEI P4? Crearemos el fichero ejemplo.tei.xml (el siguiente fragmento de texto está en UTF−8)

```
<!-- %%%%%%%%%%%%%%%%%%%%%%%%%%%% empieza ejemplo.tei.xml -->
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN"
  "/usr/local/share/sgml/tei/dtd/tei2.dtd" [
  <!ENTITY % TEI.XML          'INCLUDE' >
  <!ENTITY % TEI.dictionaries 'INCLUDE' >
  ]>

<body>
    <div0 type='dictionary'>
        <!-- English-espaÃ±ol -->
        <entry>
            <form type="abbrev">
                <orth>A</orth>
            </form>
            <form type="full">
                <lbl>abbreviation for</lbl>
                    <orth>AmpÃ¨re</orth>
                <trans>
                    <tr>A</tr>
                </trans>
            </form>
        </entry>

        <entry>
            <form>
                <orth>AmpÃ¨re</orth>
            </form>
            <gramGrp>
                <pos>n</pos>
```

```
            </gramGrp>
            <def></def>
            <trans>
                <tr>Amperio</tr>
                <gen>m</gen>
            </trans>
        </entry>

        <entry>
            <form type="abbrev">
                <orth>APL</orth>
            </form>
            <form type="full">
                <lbl>abbreviation for</lbl>
                    <orth>A Programming Language</orth>
                <trans>
                    <tr>APL</tr>
                </trans>
            </form>
        </entry>

        <entry>
            <form>
                <orth>A Programming Language</orth>
            </form>
            <gramGrp>
                <pos>n</pos>
            </gramGrp>
            <def>
                APL es la sigla de "A Programming Language" (Un Lenguaje de
                ProgramaciÃ³n), que fue un libro escrito en 1962 por el credor del
                lenguaje, Kenneth E. Iverson. Basado en lo que antes se habÃ-a conocido
                como la Nomenclatura Iverson, el APL es un lenguaje de programaciÃ³n
                extremadamente conciso, diseÃ±ado para el manejo de los arreglos
                (arrays). Los arreglos pueden ser escalares, vectoriales, tablas o
                matrices de dos o mÃ¡s dimensiones, pudiendo estar compuestos de
                informaciÃ³n numÃ©rica o alfanumÃ©rica. Bajo la conducciÃ³n de Iverson,
                IBM introdujo en 1966 el APL\360.
                Como el APL evita la introducciÃ³n de las computadoras personales,
                originalmente se lo empleÃ³ unicamente en mainframes. Desde 1983, han
                estado disponibles las versiones de APL para las PC. Debido a su
                conjunto de caracteres especiales expandidos, el APL requiere un
                teclado especial, o el uso de macros, para el ingreso de datos.
                Las versiones actuales de APL para mainframes y PC se denominan
                APL2.
            </def>
            <trans>
                <tr>Lenguaje de ProgramaciÃ³n APL</tr>
                <gen>m</gen>
            </trans>
        </entry>
  </div0>
</body>
<!-- %%%%%%%%%%%%%%%%%%%%%%%%%%% acaba ejemplo.tei.xml -->
```

¿Cómo convertir esta fuente en un documento distribuible? Utilizaremos lib−saxon−java, passivetex y las hojas de estilo XSL creadas por Sebastian Rahtz (http://www.tei−c.org/Stylesheets/teixsl.html). Lo siguiente depende de la instalación de cada uno; yo utilizo en estos momentos una Debian Sarge. Passivetex es un paquete *apt−gettable*; descargo las DTD's TEI y las hojas de estilo y las instalo en `/usr/local/share/sgml/tei/`

La conversión a fichero .pdf me da errores; teóricamente se haría

```
java −classpath /usr/share/java/saxon.jar \
com.icl.saxon.StyleSheet −o ejemplo.tei.fo ejemplo.tei.xml \
/usr/local/share/sgml/tei/xsl/fo/tei.xsl

pdfxmltex ejemplo.tei.fo
```

```
pdfxmltex ejemplo.tei.fo
```

Para generar el fichero `ejemplo.resultado.html`

```
java −classpath /usr/share/java/saxon.jar \
com.icl.saxon.StyleSheet ejemplo.tei.xml \
/usr/local/share/sgml/tei/xsl/html/teihtml.xsl
```

Este es el fichero generado:

```
<!−− %%%%%%%%%%%%%%%%%%%%%%%%%%%% empieza ejemplo.resultado.html −−>
<!DOCTYPE html
  PUBLIC "−//W3C//DTD HTML 4.0 Transitional//EN">

<div class="teidiv">
   <h2><a name="ej.utf8−div0−d0e3"></a>1.
   </h2>



                A


                abbreviation for
                    Amp&egrave;re

                    A




                Amp&egrave;re


                n



                Amperio
                m




                APL


                abbreviation for
                    A Programming Language

                    APL




                A Programming Language


                n


                APL es la sigla de "A Programming Language" (Un Lenguaje de
                Programaci&oacute;n), que fue un libro escrito en 1962 por el credor del
```

```
lenguaje, Kenneth E. Iverson. Basado en lo que antes se hab&iacute;a conocido
como la Nomenclatura Iverson, el APL es un lenguaje de programaci&oacute;n
extremadamente conciso, dise&ntilde;ado para el manejo de los arreglos
(arrays). Los arreglos pueden ser escalares, vectoriales, tablas o
matrices de dos o m&aacute;s dimensiones, pudiendo estar compuestos de
informaci&oacute;n num&eacute;rica o alfanum&eacute;rica. Bajo la conducci&oacu
IBM introdujo en 1966 el APL\360.
Como el APL evita la introducci&oacute;n de las computadoras personales,
originalmente se lo emple&oacute; unicamente en mainframes. Desde 1983, han
estado disponibles las versiones de APL para las PC. Debido a su
conjunto de caracteres especiales expandidos, el APL requiere un
teclado especial, o el uso de macros, para el ingreso de datos.
Las versiones actuales de APL para mainframes y PC se denominan
APL2.


Lenguaje de Programaci&oacute;n APL
m



</div>
<!-- %%%%%%%%%%%%%%%%%%%%%%%%%%%% acaba ejemplo.resultado.html -->
```

2.3. Correctores ortográficos

Tema relacionado con el de los diccionarios en tanto que listas de palabras.

«Nuestro» proyecto es el *COES*: Herramientas para Procesamiento de Lenguaje Natural en Español (http://www.datsi.fi.upm.es/~coes/), de Santiago Rodríguez & Jesús Carretero, que se distribuye como software de libre disposición desde finales de 1994.

COES consta de un diccionario de unos 53.000 términos y un corrector ortográfico integrado en la utilidad Unix ispell y desarrollos derivados (aspell). Hay que señalar que se ha ampliado el conjunto de herramientas lingüísticas con un diccionario de sinónimos/antónimos. Su particularidad es ser sensible a las reglas morfológicas de las palabras y no sólo a las raíces.

Un corrector ortográfico «inteligente» tiene que hacer algo más que comparar las palabras del texto con una lista de palabras correctas; para distinguir «a» de «ha» tiene que tener reglas acabaremos necesitando un mínimo análisis morfológico y sintáctico.

2.4. Correctores ortotipográficos

Estudiar la implementación de OpenOffice.

2.5. Terminologías y herramientas de gestión terminológica (TMS)

2.5.1. ¿Terminología o Memorias de traducción?

Discutido en http://www.lisa.org/sigs/phpBB/viewtopic.php?topic=22&forum=1&4

Responde Kara Warburton:

> There are many differences between Terminology and Translation Memory. The first that comes to mind is that terminology management must occur in the source language to ensure high quality source texts before a translation memory even exists. Inconsistency and inappropriate terms in the source language is an increasing and costly problem especially for global companies. And if the inconsistency is unchecked, then it breaks the TM there will be no match with previous texts which are "supposed" to be the same. A second difference is that terminology management can document features of words in a much more granular fashion

than translation memory which only records strings of text. Definitions, contexts, usage notes, references to related terms, and so forth can assist translators to select an appropriate target term when the TM does not suggest one. And if there is an exact TM match, what if the meaning is different which can frequently occur)? Then the TM match is incorrect and the translator has to think more carefully about the terms. In this case, having a terminology system can provide the translator with the required information to correct the TM. In addition, grammatical information can be recorded which can provide data to more advanced systems down the road, such as search engines and machine translation engines.

Ingrid Haussteiner lo confirma:

> I think your question is very interesting and I think from a translator's perspective this is not an either−or thing but rather a question of many ifs.
>
> If you can choose, I would implement a terminology management system first, have people test and hone their discipline and raise their awareness of the difficulties involved in using consistent terminology (even more so if it is multilingual), analyzing choices and making suggestions (descriptive terminology work).
>
> If you implement a translation memory system after some time, you might see that people slack off a bit as far as terminology work is concerned. The big argument is that the TM system integrates with the TMS.
>
> You most certainly know that EBMT (Example−Based Machine Translation) seems to be one of the buzzwords in the industry. Some translation software already support something like "assembling", others are keen to do so. So, in the future, when working with TM technology, it will be an invaluable asset to have quality terminology as the TM also assembles from there where it finds no exact or fuzzy matches.

Mark D. Childress remacha:

> I've heard this discussion at a conference or two recently. It seems to me all terminologists shudder at the thought but some have problems communicating why the sole use of translation memories is Not A Good Thing, whereas their use together with term databases is.
>
> The simplest way to explain to those people not familiar with either, but who make the decisions whether to support terminology management or not:
>
> - Translation memory: Descriptive The way things are
> - Terminology database: Prescriptive The way things should be
>
> Then ask them: Should one use a term that is (potentially) incorrect, simply because that's the way it is? Because if that's so, a term use error continues to multiply because there's no prescriptive ruling to use as a reference to standardize the term. As Kara mentioned in her post, it will ultimately make the translation memory useless.
>
> Alternately, one can correct the problem each time it appears in "reality" and count the cost of correction, whereas defining the "ideal" goes a long way towards reducing the occurrence of the problem to begin with. Basic quality rule: Get it right the first time, and reduce your follow−on costs.

2.5.2. Propuesta

El tema es complicado. Si tenemos problemas para terminar un diccionario, ¿podemos ponernos a soñar en bases de datos terminológicos? Cuando las herramientas estén maduras (pienso en los desarrollos deseables de gtranslator y kbabel) implementar bases de datos terminológicos en OLIF2 o TBX.

2.6. Correctores gramaticales y de estilo

¿Tengo que explicar que un corrector gramatical necesita una gramática?

Estudiar *diction*. Aplicarlo al español.

2.7. Corpora monolingües

2.7.1. Discusión

Según Catherine Ball (http://www.georgetown.edu/cball/corpora/):

> To create a rudimentary *concordancer* is a simple programming task: it is a matter of indexing words to lines, sorting the words alphabetically, and displaying each word in a fixed amount of context. However, most of the generally−available concordancers have many additional features, including options for producing full or partial concordances, sorting in a variety of orders, searching for collocations, and producing basic text statistics.

2.7.2. Propuesta

1. *Herramientas estadísticas*. Como traductor opino que necesitamos concordancias, colocaciones, análisis de patrones léxicos y agrupaciones de palabras

2. *Anotaciones*. Mediante etiquetado XCES si pretendemos avanzar en el campo de la traducción automática. Desarrollar/liberar herramientas de desambiguación

---

2.8. Corpora paralelos y multilingües

2.8.1. Discusión

Es evidente el interés de estas herramientas. Condiciones previas son

1. la *segmentación* y

2. el *alineado* de los textos paralelos, en una primera fase a nivel de párrafo (insuficiente)

- WA Gale y KW Church sientan las bases teóricas (estadísticas) sobre el alineado de texto en *A Program for Aligning Sentences in Bilingual Corpora*, de 1993 (http://citeseer.nj.nec.com/gale93program.html; el documento incluye la fuente del programa)

- Aportaciones del proyecto MULTEXT

- Desarrollos matemáticos a la última que escapan a mi comprensión actual,

en las páginas de Dan Melamed (http://www.cs.nyu.edu/~melamed/interests.html)

Melamed ha creado también gran número de utilidades libres.

2.8.2. Propuesta

Segmentación y alineado a nivel de párrafo están favorecidos por el marcado DocBook de nuestros documentos.

Si logramos resolver el tema del alineado de textos por otras vías en las TM no es prioritario el trabajo en este campo (pero dicen los profesionales que el alineado a nivel de párrafo no es suficiente).

2.9. Memorias de traducción

2.9.1. Discusión

Las TM contribuyen a la calidad de las traducciones en dos aspectos: su coherencia y su exactitud.

2.9.2. Propuesta

gtranslator y kbabel utilizan de forma natural MT cuando trabajan con ficheros .po.

Deben extenderse para utilizar MT cuando trabajen con documentación y textos libres. Para ello es necesario haber resuelto los problemas de la segmentación, alineación y marcado.

Ambos proyectos deben trabajar (o al menos ser capaces de importar y exportar) el formato TMX.

Notas

[1] Basta consultar la clasificación de dos páginas del término «diccionario» que hace Martínez De Sousa *Diccionario de lexicografía práctica* para llegar a la conclusión de que lo importante no es cómo se llama sino qué clase de diccionario queremos.

[2] http://cvs.hispalinux.es/cgi−bin/cvsweb/rl−dicc/.

[3] http://lemarios.olea.org

[4] (http://www.tei−c.org/Guidelines2/index.html. El documento puede descargarse comprimido como http://www.tei−c.org/Guidelines2/p4html.tar.gz