ECPC: European Comparable and Parallel Corpora
Programación básica enfocada a la confección de corpusecpc.xtrad.uji.es

Fernando Gabarrón Barrios
gabarron@uji.es

I. Resumen

El Equipo de Corpus Comparables y Paralelos de Discursos Parlamentarios Europeos (ECPC) es un grupo de investigación constituido por miembros procedentes de diferentes universidades europeas que ha recibido financiación del Ministerio de Educación y Ciencia de España (HUM2005-03756/FILO), y del Ministerio de Ciencia e Innovación (FFI2008-01610/FILO).

La investigadora principal del grupo es la Catedrática María Calzada Pérez (UJI). Está integrado por algunas de las celebridades más prestigiosas del ámbito de los Estudios Traductológicos de Corpus: la Catedrática Mona Baker, la Dra. Silvia Bernardini, la Dra. Dorothy Kenny y el Dr. Saturnino Luz, entre otras, procedentes de distintas universidades europeas. Actualmente, el grupo cuenta con la colaboración de dos jóvenes promesas de la investigación: Antonio Castro Estandía (UJI) y Fernando Gabarrón Barrios (UJI).

ECPC es un grupo multidisciplinar y plurinacional que emplea la tecnología de la información en el estudio de una de las actividades potencialmente más cooperativas de la sociedad: la traducción.

Nuestros objetivos son:

- Análisis textual, estudios traductológicos de corpus, estudios de traducción, género e ideología, y estudios descriptivos de traducción de discursos pronunciados en las sesiones plenarias de varios parlamentos europeos: el Parlamento Europeo (EP), el Congreso de los Diputados español (CD), la Cámara de los Comunes británica (HC), y próximamente el Deutscher Bundestag (DB).
- Alineado de discursos (originales y traducidos, en inglés y en español) pronunciados ante el Parlamento Europeo (estudio de corpus paralelos).
- Estudio de los discursos del Parlamento Europeo junto con los discursos de los parlamentos mencionados anteriormente con el fin de generar concordancias, listas de palabras, colocaciones, y otro tipo de información similar (estudio de corpus comparables).
- Comportamiento normativo de textos originales y traducidos.
- Desarrollo de herramientas on-line (búsquedas avanzadas, alineación, y generación de concordancias) para ponerlas a disposición de la comunidad científica y la académica.

Palabras clave: lingüística de corpus, discursos parlamentarios europeos, estudios traductológicos de corpus, didáctica de la traducción, codificación en XML, Archivo ECPC.

II. Introducción

Este artículo de investigación presenta la fase actual del proyecto ECPC. Este proyecto comenzó su labor en 2004, por este motivo, se encuentra muy desarrollado. Nuestra herramienta on-line más importante, la cual se puede consultar libremente, es el Archivo ECPC (ecpc.xtrad.uji.es); dicha herramienta aglutina la labor realizada en este proyecto durante la última década.

En este artículo se expone una breve presentación del proyecto y de sus integrantes. A continuación, se centra en el proceso de codificación en XML para poder preparar los textos originales de forma que puedan ser utilizados en nuestro Archivo para poder realizar búsquedas avanzadas. Para finalizar, se muestran ejemplos de búsquedas avanzadas con una de las herramientas on-line incluidas en nuestro Archivo: Glossa. Nuestras interfaces de búsqueda más importantes son ConcECPC, una interfaz de búsqueda de concordancias monolingüe, desarrollada por Saturnino Luz (Trinity College Dublin); y Glossa, una interfaz de búsqueda de concordancias bilingüe, desarrollada por Anders Nøklestad (Universiteter i Oslo).

Mi labor principal como becario de investigación del proyecto EPCP a lo largo del 2014 ha sido la de aplicar todo el proceso de codificación en XML de los textos originales para enriquecer el Archivo ECPC. En concreto, mi primera experiencia se ha basado en el tratamiento y codificación de los discursos del Congreso de los Diputados (CD). Durante el 2015 trabajaré con los discursos del Deutscher Bundestag (DB), así como en otros aspectos de mantenimiento y mejora del Archivo y la difusión del mismo.

Por consiguiente, este artículo se centrará en la codificación de textos de discursos parlamentarios europeos para poder finalmente realizar búsquedas avanzadas y poder llevar a cabo estudios contrastivos de ideología y género, entre otros. La codificación en este proyecto es realmente sofisticada, sin embargo, hay algunos aspectos básicos que son comunes en la confección de todo corpus.

III. Objetivos

Nuestra principal hipótesis se basa en que discursos como los del Parlamento Europeo poseen ciertas dosis de autonomía macrolingüística y/o microlingüística respecto a parlamentos nacionales europeos, debido a la limitación del grado de unificación europea.

En este proyecto se intenta comprobar o refutar esta hipótesis de trabajo a través de la creación de un Archivo electrónico compuesto por un corpus de discursos parlamentarios del Parlamento Europeo, y corpus de discursos parlamentarios de algunos de los diferentes Estados Miembros. Estos discursos están etiquetados y codificados en XML, alineados, y anotados, para poder realizar búsquedas avanzadas en nuestro servidor

ecpc.xtrad.uji.es. Ejemplos de búsquedas avanzadas serían búsquedas contrastivas por género o por ideología, sobre este aspecto se proporcionan ejemplos en el apartado 5. (Resultados).

IV. Material y método

Los requisitos mínimos del ordenador de trabajo para llevar a cabo las tareas necesarias de este proyecto son bastante asequibles, estas características mínimas son una recomendación a partir de mi experiencia de trabajo: CPU 2 x 2 GHz (doble núcleo) y RAM 2 GB. Por debajo de estas características se podrían experimentar problemas como lentitud o falta de respuesta de nuestro ordenador a la hora de procesar grandes cantidades de datos o de trabajar con algunos programas como jEdit (basado en java).

IMPORTANTE: En este proyecto se puede trabajar con Windows, Mac o Linux. Sin embargo, por motivos de comodidad y eficiencia, se recomienda estrictamente trabajar con Mac o con Linux.

Windows es una interfaz gráfica basada en el sistema operativo DOS ("anti programación"), destinada a usuarios que no necesitan programar en su rutina de trabajo. Por este motivo, si decidimos utilizar Windows, debemos instalar una serie de software que no es necesario instalar en Mac o en Linux, porque en estos ya vienen instalados todos los lenguajes necesarios de programación básica que necesitamos.

Fase 0: Instalación del software necesario:

- HTTrack http://www.httrack.com/ SiteSucker para Mac. Aplicación informática para realizar descarga total o parcial de una página web.
- Dwimperl http://dwimperl.com/windows.html Únicamente si se decide trabajar con Windows. Mac y Linux llevan por defecto el lenguaje de programación Perl instalado.
- jEdit http://www.jedit.org/ Aplicación informática basada en java para edición de textos.
- Cygwin http://www.cygwin.com/ Únicamente si se decide trabajar con Windows. Windows es una interfaz gráfica que trabaja con sistema operativo DOS; Cygwin emula un sistema operativo Unix, es decir, el que utilizan Mac y Linux.
- Filezilla https://filezilla-project.org/ Esta aplicación informática se utiliza en este proyecto para transferir datos entre el servidor XTRAD, donde está ubicado el Archivo ECPC y el ordenador de trabajo.

Fase 1: Recopilación de los corpus: textos del Parlamento Europeo (PE), de la Cámara Baja del Parlamento británico (House of Commons, HC), de la Cámara Baja del Parlamento español (Congreso de los Diputados, CD) y de la Cámara Baja del Parlamento irlandés (Dáil Éireann, DE). Tratamiento y limpieza de textos recopilados.

La descarga de textos es bastante similar para todos los parlamentos, sin embargo, el etiquetado cambiará considerablemente, así que deberemos crear un script para cada parlamento, debido principalmente a sus diferencias macrotextuales. En esta fase utilizamos el programa HTTrack para PC o SiteSucker para Mac, este software es muy intuitivo y fácil de utilizar; su utilidad es la descarga completa o parcial de páginas web. Por lo tanto, para descargar los documentos de los discursos parlamentarios, introducimos la ruta de la página web, o del directorio donde estén ubicados los discursos, y procedemos a la descarga de nuestra materia prima.

Fase 2: Etiquetado y alineamiento de los archivos recopilados en la fase anterior: básicamente este proceso consta de los siguientes pasos: experimentar con unas muestras para crear las búsquedas y reemplazos para extraer los datos de los diputados en un caso (metadatos), y en el otro para estructurar las sesiones plenarias con nuestra propia estructura xml, convertir estas búsquedas y reemplazos en one-liners de Perl, crear un script (archivo de procesamiento por lotes) donde se incluyen todos los one-liners de Perl, ejecutar en terminal el script en el directorio donde se encuentran el script y los archivos html o xml con los que estamos trabajando, crear una DTD para validar los documentos xml y su codificación, en el caso de los metadatos, crear una tabla en txt con todos los datos, en el caso de las sesiones plenarias juntar las intervenciones con la tabla de metadatos, alinear textos (no aplicable al CD), y almacenamiento de materiales en nuestro servidor ecpc.xtrad.uji.es.

A continuación se explica con detalle el proceso de codificación en XML: escogemos una serie de muestras de los textos descargados, unas 5 muestras deberían ser suficientes. Copiamos y emplazamos estos 5 textos de muestra en un determinado directorio, aparte del directorio donde se encuentran todos los textos originales. En este directorio trabajaremos experimentando y probando todo lo que sea necesario hasta crear nuestro script, así que en este directorio se puede trabajar sin miedo a cometer errores, hasta crear el script definitivo que pasaremos a todos los textos originales con los que estamos trabajando.

2.1. Creación de expresiones regulares (búsquedas y reemplazos) en jEdit. Se crea cada búsqueda y reemplazo de forma individual (se pueden ir recopilando en un archivo txt) y se prueban en las muestras de documentos con jEdit.

2.2. Cuando ya se tienen probadas las expresiones regulares en jEdit, pasamos a la **creación de un script** (programa) de Shell (.sh) para aplicar todas las expresiones regulares en todos los documentos que deseemos. Creación del script:

- Creamos un archivo con jEdit, y lo guardamos con la extensión .sh (nombre_del_archivo.sh), este es nuestro script, para este etiquetado específico del CD en este caso, ya que en este proceso se van a exponer ejemplos de dicho parlamento. La

primera línea al principio del documento indica que es un script, es de color naranja: #!/bin/sh

- También podemos insertar comentarios, que el ordenador no tendrá en cuenta a la hora de ejecutar el script. Comienzan la línea con almohadilla (#) y son de color rojo: #esto es un comentario, fecha de creación, autores, etc.
- Usaremos lenguaje de programación Perl para etiquetado de corpus (Perl es el lenguaje de programación más recomendable para trabajar con textos). Esto es una entrada de línea de comando (one-liner) de Perl de búsqueda y reemplazo vacía:

  perl -pi -e 's///g' *.xml (si trabajamos con archivos xml)

  perl -pi -e 's///g' *.html (si trabajamos con archivos html)

3.3. A continuación abrimos un archivo para **editar con jEdit**.

- Lo primero que vamos a hacer en jEdit es configurarlo para trabajar con formato de codificación de caracteres UTF-8:
- Utilities / Buffer Options / Character Encoding (UTF-8)
  Utilities / Global Options / Encodings / Default Character Encoding (UTF-8)
  Vamos a crear y probar con jEdit una serie de expresiones regulares que nos permitan buscar y reemplazar la información que nos interesa.
  Para más comodidad de visionado:
  jEdit / Utilities / Buffer Options / Word wrap (soft)
- Expresiones regulares:
  jEdit / Help / jEdit Help / Using jEdit / Regular Expressions
  (http://es.wikipedia.org/wiki/Expresión_regular)
- Ejemplos de búsqueda y reemplazo:
  # Poner todo el texto en una línea
  Find:
  (\n|\r)+
  Replace:
  (este espacio se deja en blanco)

Perl trabaja mejor si todo el texto del documento está en una línea, en este primer paso, se buscan todos los saltos de página y se reemplazan por “nada”. Ejemplos de expresiones regulares (regex): \n “salto de párrafo”, \r “retorno de carro”

Borrar información no necesaria antes de los datos del diputado
Find:
^.*<div id=“(datos_diputado)">
Replace:
XXZZ$1YYWW

En find, lo que se pone entre paréntesis, equivale a $1, $2, $3, etc., en el replace; estos $ se pueden emplazar en el orden que queramos, u omitir, si es necesario.

Ejemplos de regex:

^ el acento circunflejo equivale a principio de línea.

. el punto equivale a cualquier carácter.

* el asterisco equivale al carácter anterior 0 veces, 1 vez, o varias veces repetido.

Estas expresiones regulares las utilizaremos posteriormente en los one-liners (entre la primera y la segunda barra pegamos la búsqueda y entre la segunda y la tercera pegamos el reemplazo).

Ejemplos de one-liners, basados en los ejemplos anteriores:

```
perl -pi -e 's/(\n|\r)+//g' *.html
perl -pi -e 's/^.*<div id="(datos_diputado)">/XXZZ$1YYWW/g' *.html
```

En el script, se puede observar como el contenido entre las barras (find and replace) es de color rosa, esto indica que el one-liner con Perl está correcto; el principio y el final de la línea quedan de color negro.

Es muy importante a la hora de trabajar con perl conocer el conjunto de metacaracteres para expresiones regulares:

\ ^ $. [ ] { } | ( ) * + ?

Estos metacaracteres, en una expresión regular, son interpretados en su significado especial y no como los caracteres que normalmente representan. Una búsqueda que implique alguno de estos caracteres obligará a "escaparlo" de la interpretación mediante \, como se hace para evitar la interpretación por el shell de los metacaracteres del shell.

Ejemplos:

En una expresión regular, el carácter . (punto) representa "un carácter cualquiera"; si escribimos \. , estamos representando el carácter . (punto) tal cual, sin significado adicional.

En una expresión regular, los caracteres ( ) representan "principio y final de un reemplazo $"; si escribimos \(y \) , estamos representando los caracteres tal cual, simplemente paréntesis, sin significado adicional.

Una vez tenemos los html con los datos de los diputados y las sesiones plenarias, vamos a buscar los metadatos que nos interesan en sus html individuales, y a definir nuestra macroestructura para las sesiones plenarias. Los datos diputados al final se recopilarán en un solo documento .txt. Las sesiones plenarias las transformaremos en .xml (html usa unas etiquetas determinadas, mientras que con xml podemos usar nuestras propias etiquetas). Tenemos dos opciones, podemos abrir los html de uno en uno y apuntar los datos manualmente de los diputados, así como definir nuestra macroestructura en las sesiones plenarias “manualmente”; o podemos programar en Perl, y crear una serie de one-liners (entradas de línea de comando) para lograr nuestro objetivo mucho más rápida y

cómodamente. De esta forma podremos procesar archivos por lotes, por ejemplo 100, 1.000, etc. a la vez.

En Linux y Mac, el terminal está preparado para programar en Perl y otros lenguajes de programación, pero en Windows no. Por eso los usuarios de Windows necesitan instalar Cygwin para emular un terminal de Linux o Mac. Es muy importante durante la instalación de Cygwin tener en cuenta un detalle a la hora de instalar los paquetes; por defecto no los instala todos de forma completa, así que tenemos que intervenir en la instalación de los paquetes para que se instale todo lo que nos interesa. De todas formas, si queremos instalar más paquetes posteriormente, solo tenemos que ejecutar el instalador de nuevo, y en la pantalla de paquetes elegir los que queremos instalar.

Por lo tanto, ejecutamos para instalar Cygwin, le damos a todo *Next*, hasta una pantalla donde nos da a elegir un *Mirror* para descargar (elegimos por ejemplo http://cygwin.mirror.constant.com); hacemos click en siguiente hasta la pantalla de Paquetes. En todos pone *Default*, porque solo instala lo mínimo para funcionar. Nos interesa instalar Perl, Python y Shells (completos), hacemos click en *Default* en cada uno de estos paquetes, hasta que ponga *Install*, y procedemos a la instalación.

Ahora ya tenemos instalado Cygwin y preparado para trabajar con Perl y Shell. Algunos ejemplos de órdenes básicas de la terminal Cygwin (son los mismos que en Mac o Linux):

Introducción a Cygwin (sistema operativo Unix):

# Acceder a los directorios con cygdrive	$ cd /cygdrive
# Ver el directorio actual (present working directory)	$ pwd
# Ver el directorio actual de Cygdrive	$ pwd /cygdrive
# Listar (list)	$ ls
# Cambiar directorio (change directory)	$ cd
# Subir un nivel de directorio	$ cd ..

Más comandos para Unix:

http://www.linuxtotal.com.mx/?cont=info_admon_002

Acceder a un directorio donde vamos a trabajar, ejemplo:

cd /cygdrive/c/Users/mc/Desktop/NEW_FOLDER

2.3. Ejecución de órdenes individualmente: En esta prueba se muestra como ejecutar una orden de Perl (Find and Replace) de forma individual. Nuestro objetivo es aglutinar muchas órdenes de Perl en one-liners dentro de un único script, para ejecutarlas todas a la vez.

Creamos un directorio por ejemplo en el escritorio, en C: sería más rápido y cómodo para trabajar con Cygwin, en Linux o Mac, simplemente hacemos click con el botón derecho en un directorio, y escogemos la opción “abrir un terminal”. En este directorio introducimos 5 archivos html de 5 diputados que usaremos en esta prueba. Los archivos html con los que trabajaremos, 5 en este caso, deben estar al nivel del directorio donde vamos a trabajar, y no en un subdirectorio.

1. Abrimos el terminal en el directorio en el que estamos trabajando, por ejemplo:

$ cd /cygdrive/c/Users/mc/Desktop/NEW_FOLDER

2. Abrimos el documento .sh en jEdit
3. Copiamos la primera orden
   perl -pi -e 's/(\n|\r)+//g' *.html
4. La pegamos en el teminal así (Shift + Insert)
   $ perl -pi -e 's/(\n|\r)+//g' *.html
5. La ejecutamos (Enter) y en la siguiente línea de la terminal no aparece nada, lo comprobamos en jEdit y vemos como la orden se ha ejecutado en los html y hemos cambiado lo que queríamos.

Si al ejecutar una orden con la terminal, en la siguiente línea no aparece nada, significa que la operación se ha llevado a cabo con éxito; en caso contrario, la terminal nos marcaría los errores que hemos cometido.

2.4. Ejecución de varias órdenes (procesamiento por lotes): A continuación añadimos a este directorio también el .sh que hemos creado con jEdit, al mismo nivel que los html, no en un subdirectorio.

1. Abrimos el terminal en el directorio en el que estamos trabajando, por ejemplo:
   $ cd /cygdrive/c/Users/mc/Desktop/NEW_FOLDER
2. Ejecutamos el script, que contiene todos los one-liners que hemos probado de uno en uno, por ejemplo:
   $ sh mp_cd_regex.sh *.html
   ("sh" significa que le ordenas al ordenador que use una orden de shell, "mp_cd_regex.sh" es el script que hemos creado con datos diputados, y "*.html" significa que le ordenamos al ordenador que lo aplique en todos los archivos con extensión .html).
3. En la siguiente línea no aparece nada, pero lo comprobamos en jEdit y vemos como el script se ha ejecutado en los todos los html de ese directorio y hemos cambiado lo que queríamos.

Si al ejecutar, en la siguiente línea no aparece nada, es que la operación se ha producido con éxito; en caso contrario, el terminal nos marcaría los errores que hemos cometido.

2.5. Creación de un archivo de validación (DTD): Parecido al corrector ortográfico de los procesadores de texto, pero en vez de faltas de ortografía, señala fallos en la codificación del xml, lo que significa que si no se corrigen estos errores, nuestros archivos xml no funcionarán correctamente en nuestros buscadores on-line.

Para validar hace falta instalar dos pluggins en jEdit:

jEdit / Pluggins / Pluggin Manager

Seleccionar e instalar estos dos:

- JDiffPluggin

- XML

Al seleccionar estos pluggins para instalarlos, otros pluggins se seleccionarán automáticamente (no tocar). Click en *Install* (abajo a la izquierda).

Generación de una DTD (Definición de Tipos de Documento):

jEdit / Pluggins / XML / Generate DTD

Ahora podemos validar nuestros documentos XML (comprobar que todas las etiquetas están correctas, es decir, en su lugar correcto, y con apertura y cierre). Así queda en el encabezamiento de nuestro script, el nombre del .dtd que tendremos en nuestro directorio debe ser igual al del encabezamiento de nuestro documento. Ejemplo:

<\?xml version="1\.0" encoding=\"UTF-8\" standalone="no"\?>\n<\!DOCTYPE ecpc_EP SYSTEM "mp_CD.dtd">

Al abrir nuestros documentos XML, se abrirá una ventana con un "Error List", ahí aparece el error y la línea donde se encuentra dicho error.

Otra opción es fijarse en las líneas subrayadas en rojo, este subrayado marca errores de etiquetado de forma parecida a los procesadores de texto, como por ejemplo LibreOffice, cuando subrayan en rojo errores gramaticales. De esta forma podemos comprobar los documentos de uno en uno. El siguiente paso sería comprobar todos los documentos de un directorio a la vez:

Validación de los documentos en XML y su codificación, para así comprobar que está todo bien. Aplicar el comando en terminal:

$ xmllint -valid -noout *.html

Si nuestros documentos fueran xml sería:

$ xmllint -valid -noout *.xml

Para validar solo un documento de forma individual en el terminal:

$ xmllint -valid -noout nombrearchivo.xml

Creación de una tabla con datos diputados, pegado de documentos y a la vez transformación en xml

$ cat *.html > total.xml

Si hay problemas con Cygwin, debemos instalar en Packages/(search) RCS (e instalar este paquete para poder juntar varios documentos en uno único).

Una vez todo junto y sin espacios, copiamos todos los diputados y los pegamos en una hoja de excel. Por comodidad podemos ordenamos alfabéticamente.

Una vez transformados en xml, le cambiamos la extensión a .txt a la tabla final que contiene los datos diputados y usaremos un script confeccionado por Saturnino Luz para el juntado de datos de oradores en cada intervención de cada orador.

Finalmente procedemos a alinear los textos originales y los traducidos, aunque este paso no es aplicable a los discursos de los parlamentos monolingües, es un paso muy importante para los discursos del Parlamento Europeo. La herramienta que poseemos para el alineado es Intertext, desarrollada por Pavel Vodrinka, y está ubicada en nuestra interfaz gráfica on-line en ecpc.xtrad.uji.es.

Fase 3: Desarrollo de parámetros (macrotextuales y microtextuales) **para el análisis contrastivo de los corpus recopilados, etiquetados y alineados.**

Fase 4: Estudios contrastivos a partir de las herramientas y los resultados de las tres etapas anteriores: Publicación de los resultados a través de diferentes modalidades de difusión, diseño de una página web en la que se recoja el trabajo y resultados parciales y totales del proyecto, creación de una interfaz de consultas contrastivas siguiendo la estela de TransSearch.

V. Resultados

Durante mi experiencia en este proyecto como becario de investigación, mi tarea principal ha sido la recopilación, tratamiento y codificación en XML de textos originales para el posterior enriquecimiento de nuestro Archivo.

A continuación se pueden observar algunas capturas de pantalla que muestran un fragmento de texto original de discurso del Congreso de los Diputados extraído en formato pdf de página oficial: http://www.congreso.es (Figura 1), y el mismo fragmento codificado en XML y con los metadatos de los oradores añadidos, para su posterior utilización en búsquedas avanzadas en los diferentes motores de búsqueda que ofrece el proyecto, es decir, búsquedas por género, partido político, circunscripción, institución a la que pertenece el orador, etc. (Figura 2).

La señora **VICEPRESIDENTA** (Chacón i Piqueras): Muchas gracias, señor Herrera.

Continuamos con el Grupo Parlamentario de Coalición Canaria. Tiene la palabra el señor Rivero.

El señor **RIVERO BAUTE:** Señora presidenta, señor presidente del Gobierno, señoras y señores diputados, Coalición Canaria va a apoyar la convocatoria del referéndum y rotundamente va a implicarse en pedir el sí y el apoyo al Tratado constitucional europeo, que se va a someter a referéndum en los distintos Estados miembros. Vamos a apoyar la convocatoria del referéndum porque fue Coalición Canaria quien en el último debate del Estado de la nación, celebrado el pasado año, presentó una propuesta de resolución en ese sentido. Mi grupo parlamentario presentó una propuesta de resolución dirigida a que una vez terminada las negociaciones entre los Estados miembros se sometiera a consulta del pueblo español la propuesta de resolución que, por cierto, fue aprobada por unanimidad por la Cámara. Vamos a pedir el sí a la Constitución europea por diversas razones: primero, porque es un avance (seguramente para algunos será la botella medio vacía y para otros la botella medio llena); sin ninguna duda es un avance en el proceso de construcción de

Figura 1. Fragmento de discurso del Congreso de los diputados extraído en formato pdf de la página oficial: http://www.congreso.es

```
<intervention id='in17'>
<speaker>
<name>Chacón Piqueras, Carme</name>
<birth_date>19710313</birth_date>
<birth_place country="ES">Esplugues de Llobregat (Barcelona)</birth_place>
<status>NA</status>
<gender>female</gender>
<institution>
<ni country="ES">CD</ni>
</institution>
<constituency country="ES" region="Barcelona"/>
<affiliation>
<national_party>Partido Socialista Obrero Español</national_party>
<cd group="GS"/>
</affiliation>
<post>VICEPRESIDENTA</post>
</speaker>
<speech id='sp17' language="ES">
Muchas gracias, señor Herrera.
Continuamos con el Grupo Parlamentario de Coalición Canaria. Tiene la palabra el
señor Rivero.
</speech>
</intervention>

<intervention id='in18'>
<speaker>
<name>Rivero Baute, Paulino</name>
<birth_date>19520211</birth_date>
<birth_place country="ES">El Sauzal (Santa Cruz de Tenerife)</birth_place>
<status>NA</status>
<gender>male</gender>
<institution>
<ni country="ES">CD</ni>
</institution>
<constituency country="ES" region="Santa Cruz de Tenerife"/>
<affiliation>
<national_party>Coalición Canaria</national_party>
<cd group="GCC-NC"/>
</affiliation>
<post>NA</post>
</speaker>
<speech id='sp18' language="ES">
Señora presidenta, señor presidente del Gobierno, señoras y señores diputados,
Coalición Canaria va a apoyar la convocatoria del referéndum y rotundamente va a
implicarse en pedir el sí y el apoyo al Tratado constitucional europeo, que se
va a someter a referéndum en los distintos Estados miembros. Vamos a apoyar la
convocatoria del referéndum porque fue Coalición Canaria quien en el último
debate del Estado de la nación, celebrado el pasado año, presentó una propuesta
de resolución en ese sentido. Mi grupo parlamentario presentó una propuesta de
resolución dirigida a que una vez terminada las negociaciones entre los Estados
miembros se sometiera a consulta del pueblo español la propuesta de resolución
que, por cierto, fue aprobada por unanimidad por la Cámara. Vamos a pedir el sí
a la Constitución europea por diversas razones: primero, porque es un avance
(seguramente para algunos será la botella medio vacía y para otros la botella
medio llena); sin ninguna duda es un avance en el proceso de construcción de
```

Figura 2. El mismo fragmento de texto de la figura 1, pero codificado en XML y con los metadatos de los oradores añadidos, para su posterior utilización en búsquedas avanzadas en los diferentes motores de búsqueda que ofrece el proyecto, es decir, búsquedas por género, partido político, circunscripción, institución a la que pertenece el orador, etc.

Las siguientes capturas de pantalla muestran el motor de búsquedas bilingües Glossa, incluido en la interfaz gráfica on-line del proyecto: ecpc.xtrad.uji.es. En este caso los resultados de las búsquedas se extraerán del corpus paralelo y comparable del Parlamento Europeo. La introducción de estas capturas de pantalla en este artículo pretende aportar ejemplos de búsquedas avanzadas. En estos ejemplos se usan dos tokens juntos “gender equality” (Figuras 3, 4, 5, 6, 7).

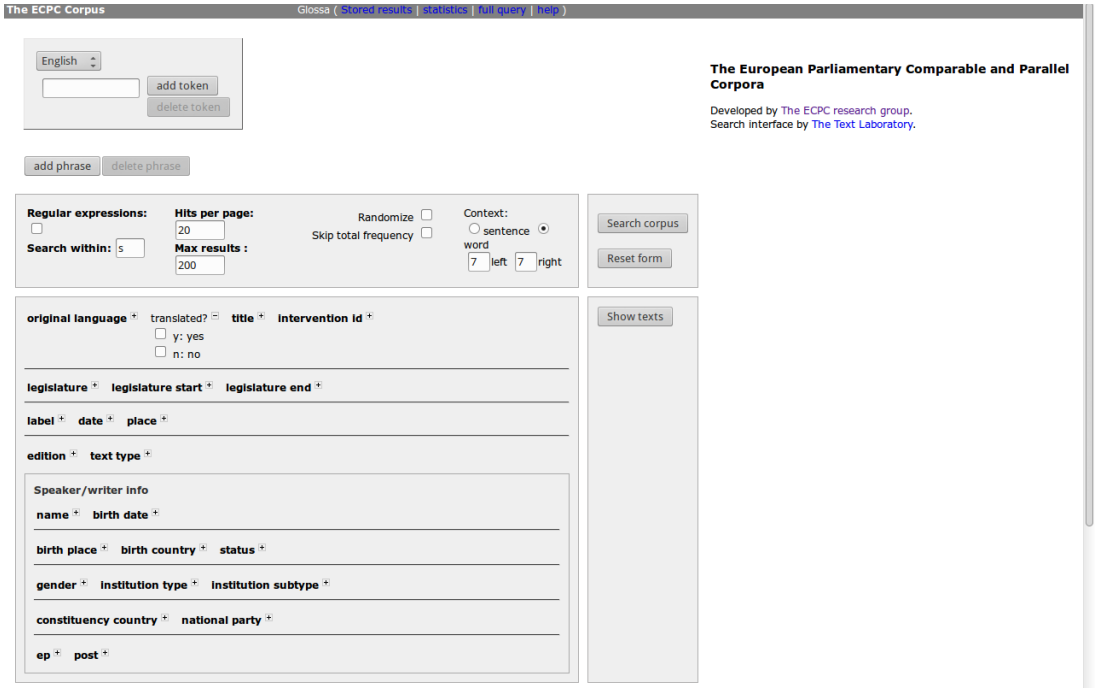

Figura 3. Interfaz gráfica de Glossa, donde se pueden observar las diferentes opciones de búsqueda.

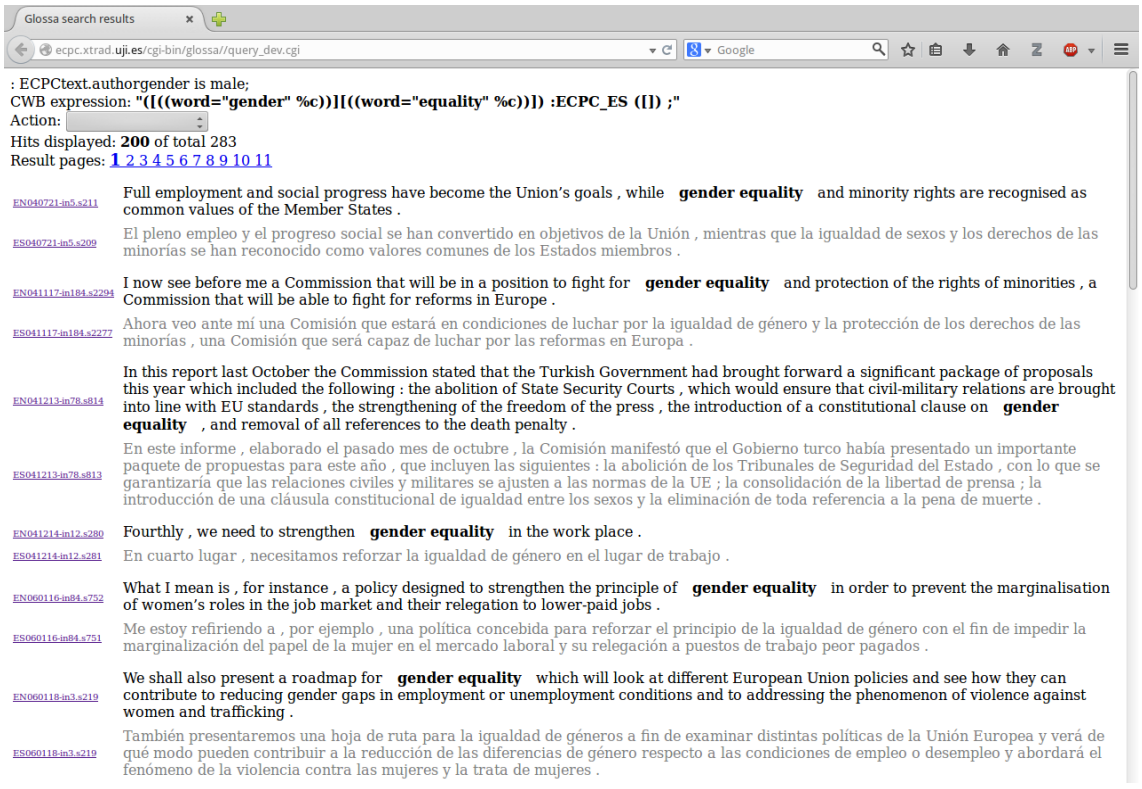

Figura 4. Búsqueda por género masculino, tokens "gender equality", 283 resultados.

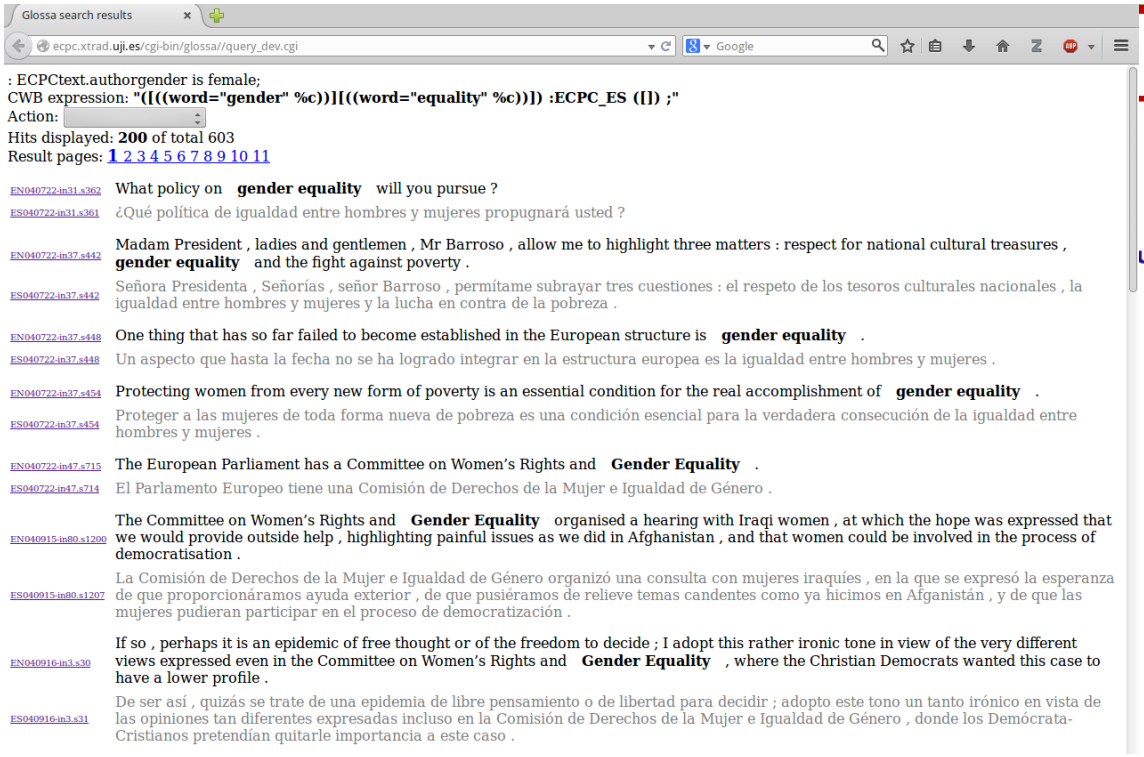

Figura 5. Búsqueda por género femenino, tokens "gender equality", 603 resultados.

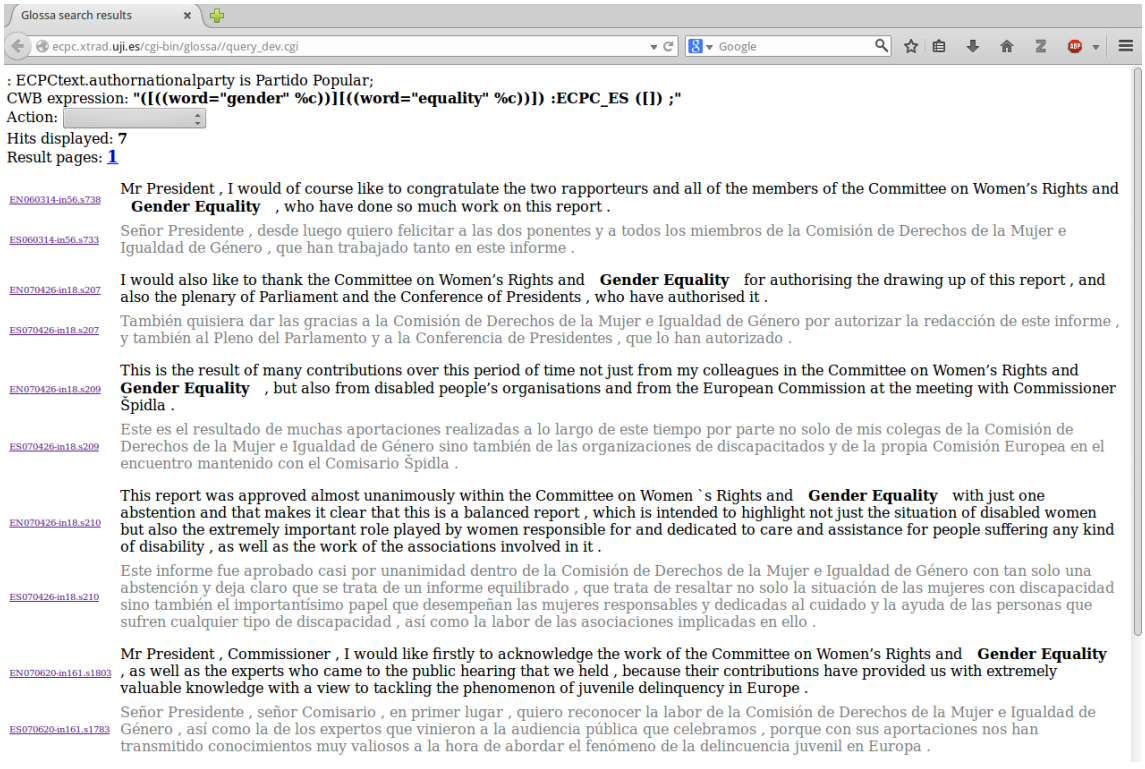

Figura 6. Búsqueda por partido político Partido Popular, tokens "gender equality", 7 resultados.

Figura 7. Búsqueda por partido político Partido Socialista Obrero Español, tokens "gender equality", 40 resultados.

En estos ejemplos expuestos con capturas de pantalla se pueden alcanzar varias conclusiones de ideología y género, como por ejemplo que las oradoras (603 resultados) utilizan más la expresión "gender equality" que los oradores (283 resultados) o que los diputados del Partido Socialista Obrero Español (40 resultados) utilizan más dicha expresión que los diputados del Partido Popular (7 resultados).

Otro aspecto a estudiar es la diferencia entre el discurso pronunciado en las diferentes cámaras parlamentarias europeas de los diferentes Estados Miembros y el mismo discurso pronunciado en el Parlamento Europeo.

La coordinadora del proyecto, María Calzada, se encarga de llevar a cabo los pertinentes estudios descriptivos y contrastivos de traducción, género e ideología, a partir de las herramientas y los resultados de las diferentes etapas del proyecto. Uno de los aspectos más importantes que se pretende analizar es el uso impreciso del lenguaje en política, como por ejemplo el abuso de significantes vacíos y/o eufemismos.

Para acceder a la interfaz gráfica on-line del proyecto ECPC: ecpc.xtrad.uji.es

Para cualquier duda sobre este artículo de investigación centrado en la codificación y tratamiento de textos en XML: gabarron@uji.es

VI. Discusión y Conclusiones

Las aplicaciones más importantes del Archivo ECPC son la aplicación de la ingeniería informática y la programación básica en la didáctica de la traducción y la investigación traductológica. Nuestras herramientas on-line ConcECPC y Glossa (ecpc.xtrad.uji.es), incluidas en nuestro Archivo ECPC on-line, están a disposición de la comunidad científica y del alumnado de traducción, así como de otros perfiles como alumnado de ciencias políticas, o cualquier persona que quisiera ampliar sus conocimientos sobre el género del discurso parlamentario en general y sobre las diferencias entre el discurso del Parlamento Europeo y el discurso parlamentario de los diferentes Estados Miembros.

VII. Bibliografía

BAKER, M. (2004): «*A corpus-based view of similarity and difference in translation*», International Journal of Corpus Linguistics, 9: 167-193.

CALZADA, M. Y MARÍN, N. Y MARTÍNEZ, J. (2006): *ECPC: European Parliamentary Comparable and Parallel Corpora / Corpus Comparables y Paralelos de Discursos Parlamentarios Europeos*, Procesamiento del lenguaje natural, ISSN 1135-5948, 37: 349-350.

CALZADA, M. Y LUZ, S. (2006): *ECPC: Technology as a tool to study the (linguistic) functioning of national and transnational European parliaments*, Journal of Technology, Knowledge and Society, 5: 53-62.

LUZ, S. (2000): *A software toolkit for sharing and accessing corpora over the Internet,* En M Gavrilidou G Carayannis S Markantonatou S. Piperidis, y G. Stainhauer, editores, Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, Athenas, Greece, Mayo. European Language Resources Association (ELRA).

NESSELHAUF, N. (2005): *Corpus Linguistics: a Practical Introduction,* Heidelberg Universität.

ORWELL, G. (1946): *Politics and the English Language*, Horizon, London.

SERRAT, I. Y MARTÍNEZ, J. (2012): *ECPC: el discurso parlamentario europeo desde la perspectiva de los estudios traductológicos de corpus,* Linguamática, ISSN: 1647-0818, 4 (2): 65-73.