Estudios Filológicos
ISSN: 0071-1713
efil@uach.cl
Universidad Austral de Chile
Chile

Hernández Osuna, Sergio; Ferreira Cabrera, Anita
Procesamiento semántico automático, enfocado en la coherencia textual, para apoyar la producción escrita de noticias
Estudios Filológicos, núm. 58, noviembre, 2016, pp. 97-122
Universidad Austral de Chile
Valdivia, Chile

Procesamiento semántico automático, enfocado en la coherencia textual, para apoyar la producción escrita de noticias*

Automatic semantic processing, focusing on textual coherence, to support the production of written news

Sergio Hernández Osuna[1], *Anita Ferreira Cabrera*[2]

[1] Universidad de Concepción, Departamento de Comunicación Social, Concepción, Chile, e-mail: sergihernandez@udec.cl

[2] Universidad de Concepción, Departamento de Español, Concepción, Chile, e-mail: aferreir@udec.cl

El siguiente artículo presenta el diseño e implementación de un módulo de análisis semántico, enfocado en la predicción de la coherencia textual, programado en Python 3. Las etapas de implementación comprenden el trabajo realizado en el diseño de una herramienta para la recopilación automática del corpus (noticias sobre política), de otra destinada a preparar los textos reunidos para su procesamiento posterior, hasta llegar al diseño de la herramienta final que realiza el análisis de los textos. El método empleado para esto es el Análisis Semántico Latente. El artículo concluye con la presentación de los resultados de las pruebas realizadas, con el fin de testear la herramienta mediante el procesamiento de textos, para observar su sensibilidad en la evaluación de la coherencia textual.

Palabras clave: Procesamiento de Lenguaje Natural, coherencia, Análisis Semántico Latente, noticias políticas.

The following article presents the steps to build a semantic analysis module focused on the prediction of textual coherence, programmed in Python 3. The steps described include the work done in the design of a tool for automatic recopilation of the corpus (politic news), another destined to prepare the texts collected for further processing, up to the final design tool that performs the analysis of the texts. The method used for this is the Latent Semantic Analysis. Finally, the article presents the results of tests performed in order to test the tool, through texts processing, with the goal of watching sensitivity in the evaluation of textual coherence.

Key words: Natural Language Processing, coherence, Latent Semantic Analysis, politic news.

* El estudio en procesamiento de lenguaje natural que se presenta en este artículo se ha desarrollado en el contexto del proyecto de investigación FONDECYT 1140651 en lo que compete a los avances del Sistema Tutorial Inteligente para mejorar la precisión lingüística (Investigadora Responsable: Dra. Anita Ferreira Cabrera).

1. Introducción

El Procesamiento del Lenguaje Natural (NLP por sus iniciales en inglés, *Natural Language Processing*), como interdisciplina de la Lingüística y la Inteligencia Artificial, es un área que está en constante progresión gracias a los avances tecnológicos en ambas disciplinas. En este escenario, el trabajo de los investigadores consiste no solo en actualizar sus propios conocimientos, como en cualquier área, sino que también es fundamental actualizar los procedimientos y herramientas gracias a las nuevas posibilidades que permite el desarrollo tecnológico, con el fin de ir respondiendo a las exigencias que este progreso constante supone.

El presente artículo describe la construcción de un analizador automático de coherencia textual, empleando fundamentalmente *scripts* programados en Python 3 y la técnica de Análisis Semántico Latente. Para el caso específico que se presenta, el analizador automático de coherencia textual se enfocó en noticias sobre política, ya que por exigencia del Análisis Semántico Latente, los textos que incluya el corpus deben pertenecer a un dominio temático restringido. El corpus con que se trabajó lo forman 7165 noticias obtenidas desde el sitio en Internet del diario *La Tercera* de Chile, desde agosto de 2012 a febrero de 2014. En el artículo se detallará el proceso realizado, partiendo desde el desarrollo de un método para la recopilación automática de los textos, pasando por la creación del *script* que realice la tarea de prepararlos para ser utilizados por la aplicación de Análisis Semántico Latente, hasta la construcción de un prototipo de analizador semántico, que lleve a cabo el procesamiento también en forma automática. Por último, el artículo presenta los resultados de las pruebas realizadas con el fin de testear la herramienta, mediante el procesamiento de textos, para observar su sensibilidad en la evaluación de la coherencia textual.

La construcción de este módulo de procesamiento semántico, enfocado en la evaluación de la coherencia textual, se enmarca en una tarea mayor de diseñar un Sistema Tutorial Inteligente (STI)[1] destinado a apoyar la producción escrita de noticias en estudiantes de periodismo. Martín Vivaldi (1993: 369) define la noticia como "el género periodístico por excelencia que da cuenta de un modo sucinto pero completo, de un hecho actual o actualizado, digno de ser conocido y divulgado, y de innegable repercusión humana". Martínez Albertos (2004: 288) la conceptualiza como el género periodístico "más escueto, más descarnado, más fuertemente ceñido al puro esqueleto del hecho o acontecimiento que se quiere transmitir. Es, diríamos, el género periodístico más rigurosamente objetivo en su propósito teórico y desde el punto de vista de la apariencia formal del lenguaje utilizado por el periodista reportero". De ambas definiciones se puede colegir que la noticia es el texto base del periodismo y que da cuenta de un modo sucinto, carente de adornos, de un hecho actual. Por lo mismo, la noticia es el texto más importante en la formación de pregrado de un estudiante de Periodismo. Y es un texto complejo de producir para los estudiantes, ya que exige comprender el hecho al que se refiere y tener la capacidad de resumirlo

[1] Un Sistema Tutorial Inteligente es "un programa para la enseñanza-aprendizaje basado en el computador cuya finalidad última es la facilitación de los procesos de aprendizajes personalizados y autónomos. Estos sistemas son capaces de comportarse como un experto, tanto en el dominio de conocimiento que enseña como en el dominio metodológico, donde es capaz de diagnosticar la situación en la que se encuentra el estudiante y, de acuerdo con ello, ofrecer una acción o solución que le permita progresar en el aprendizaje" (Ferreira, 2004; Ferreira, Salcedo, Kotz y Barrientos, 2012).

a su médula, para transmitirlo de la forma más escueta posible, sin despojarlo de la información fundamental que involucra. Por ello, se considera que la construcción de un STI para apoyar la enseñanza del proceso de escritura de noticias, será un aporte para la comunidad académica, al enfocarse tanto en la producción de la estructura semántica de la noticia, como en su precisión lingüística.

Dos son los objetivos del trabajo que se presenta, por una parte, contribuir en la investigación sobre el procesamiento semántico automático en nuestra lengua (español) y, por otra, diseñar e implementar una herramienta que evalúe la coherencia textual de noticias, que automatice la mayor parte del proceso y que sea eficiente, con el fin de integrarla en el STI que se construirá. La perspectiva empleada para enfocar el problema es claramente multidisciplinar, como se verá en el desarrollo del artículo, ya que se aborda el problema desde un punto de vista lingüístico y, también, informático.

2. Conceptos fundamentales: el análisis semántico latente como herramienta para la medición de coherencia textual

Si bien el presente trabajo busca ser fundamentalmente la descripción de un proceso, es necesario comprender qué se entenderá por coherencia textual y Análisis Semántico Latente, y sobre todo, cómo este último permite medir en forma automática la coherencia en un texto.

2.1. Coherencia

La coherencia, normalmente, se relaciona en la literatura con el concepto de cohesión y la discusión sobre los significados de ambos es amplia. Cuando De Beaugrande y Dressler (1997) plantean las siete normas de textualidad, incluyen entre estas a la cohesión y la coherencia. Señalan que la *cohesión* se refiere a la estabilidad de un texto que se mantiene gracias a la continuidad de los elementos que lo conforman. Esta "noción de continuidad se basa, a su vez, en la suposición de que existe una relación entre los diferentes elementos lingüísticos que configuran el texto". De lo anterior, se entiende que la coherencia se construye en base a los elementos que se encuentran en la superficie del texto, es decir, puede entenderse como algo, en algún sentido, *tangible*. Los mismos autores, en relación a la coherencia, indican:

> Un texto "tiene sentido" porque el conocimiento activado por las expresiones que lo componen va construyendo, valga la redundancia, una continuidad de sentido. Cuando los receptores detectan la ausencia de continuidad, el texto se convierte en un «sinsentido», característica normalmente atribuible a la existencia de una serie de desajustes entre la organización de los conceptos o de las relaciones expresadas en el texto y el conocimiento previo del mundo que tienen los receptores (De Beaugrande y Dressler, 1997: 135).

Visto lo recién expuesto, se puede afirmar que la coherencia alude a algo que va más allá de lo que se encuentra en la superficie del texto, ya que es un juego entre el texto mismo con los conocimientos de mundo que tiene el lector, por lo que a diferencia de la cohesión, para De Beaugrande y Dressler, la coherencia alude a elementos que se podrían entender como *intangibles*.

Halliday y Hasan (1976) presentan una visión de ambos conceptos un tanto diferente, ya que para ellos la *cohesión* y la *coherencia* son complementarias. La primera permite explicar las relaciones que se dan en el discurso, esto es, se refiere a los recursos (vínculos de cohesión) que permiten relacionar una oración con otras presentadas en el texto previamente, y a través de estos vínculos de cohesión (referencia, sustitución, elipsis, conjunción o el léxico) se establecen relaciones semánticas en el texto que permiten lograr la segunda, es decir, la coherencia. En palabras de Lowerse (2004), Halliday y Hasan "usan el término *coherencia* para referirse a una 'interrelacionalidad' global en el texto, mientras que reservan *cohesión* para unidades lingüísticas menores en el texto".

Álvarez señala que ambos conceptos se relacionan, pero no significan lo mismo: la cohesión es el aspecto formal, gramatical de las relaciones que existen de una oración a otra en el texto y la coherencia designa el aspecto mental, conceptual de la relación que se postula entre los hechos denotados (1995: 98). Relacionado con esto, hay que agregar que los psicolingüistas consideran que "la coherencia es más bien una función cognitiva; dicho de otro modo, la coherencia está en la mente y no en el texto" (De Vega et al., 1999: 273). De esta forma, el lector de un texto entiende las relaciones de coherencia entre las diversas partes de este gracias, en gran medida, a sus conocimientos pragmáticos del mundo. O sea, "la coherencia es el resultado de la interacción entre el texto y el saber sobre el mundo" que comparten el autor del texto escrito como quien lo lee, señala Álvarez (1995: 98), en una postura que sigue a De Beaugrande y Dressler.

Si bien la discusión en la literatura sobre ambos términos es inmensa y no está cerrada, para los fines del presente trabajo, enmarcado en el campo de la Lingüística Computacional, se asumirá la definición de coherencia textual propuesta por Jurafsky y Martin (2008). Para ellos, coherencia y cohesión son términos a menudo confundidos. La cohesión se refiere a la forma en que las unidades textuales son enlazadas. Indican que una relación cohesiva es como una clase de pegamento que agrupa dos unidades en una sola unidad mayor. Por otro lado, para los citados autores la coherencia alude a las relaciones de significados entre dos unidades. Una relación de coherencia explica cómo el significado de diferentes unidades textuales puede combinarse para construir un significado discursivo mayor (Jurafsky y Martin 2008: 605). La anterior definición se ajusta perfectamente a la técnica de Análisis Semántico Latente que se empleará en el presente trabajo y que se explica en el apartado siguiente.

2.2. Análisis Semántico Latente

El Análisis Semántico Latente (o LSA, según sus iniciales en inglés, *Latent Semantic Analysis*) es una técnica que utiliza procesos estadísticos para extraer y representar los significados de las palabras. Los significados son representados en términos de su semejanza con otras palabras en un extenso corpus de documentos (Boonthum, Levinstein y McNamara, 2007: 95). Landauer, Foltz y Laham (1998) definen al Análisis Semántico Latente como una técnica matemático-estadística, totalmente automática, para extraer e inferir relaciones de uso contextual esperado de palabras en pasajes del discurso. No es un procesamiento del lenguaje natural tradicional o programa de inteligencia artificial; no utiliza diccionarios construidos por humanos, ni bases de conocimiento, ni redes semánticas, ni gramáticas, ni

analizadores sintácticos ni morfologías o similares; y toma como única entrada el texto segmentado en palabras, definidas como cadenas de caracteres únicos y separadas en pasajes significativos o muestras como frases o párrafos[2].

McCarthy, Briner, Rus y McNamara (2007) señalan que el Análisis Semántico Latente está basado en la idea de que las palabras (o grupos de palabras) aparecen en algunos contextos pero no en otros. La ventaja práctica del LSA sobre otros tipos de mediciones como *shallow word overlap* es que va más allá de las similitudes léxicas tales como silla/sillas o correr/corrió; el LSA es capaz de evaluar la similitud semántica entre términos tales como silla/mesa, mesa/madera y madera/bosque. De este modo, el Análisis Semántico Latente no solo indica que dos términos están relacionados, sino que también señala cuán similares son (McCarthy *et al.* 2007: 111).

Landauer (2002), para explicar el funcionamiento del LSA, señala que el significado de un pasaje de texto está contenido en sus palabras, y que todas sus palabras contribuyen a formar su significado. A esto, agrega, que dos pasajes, aunque tengan diferentes palabras, podrían tener un significado similar. Esto se puede resumir asumiendo que el significado de un pasaje de texto es igual a la suma del significado de las palabras que lo componen. El autor lo representa con el siguiente esquema:

significado de la palabra 1 + significado de la palabra 2 + ...+ significado de la palabra n = significado del pasaje

Según expresan Foltz, Kinstch y Landauer (1998), el poder del Análisis Semántico Latente para determinar el grado de relación semántica proviene del análisis de un extenso número de textos de ejemplo. Para una idea más completa sobre el LSA, se recomienda revisar Venegas (2003).

2.3. La medición de coherencia textual mediante Análisis Semántico Latente

Para realizar la predicción de coherencia, el LSA representa cada palabra del texto procesado –así como unidades textuales mayores– en el espacio semántico a través de vectores. Las unidades textuales mayores –como una oración– son representadas en este espacio semántico multidimensional como el promedio ponderado de los vectores de los términos que contienen. La relación semántica entre dos unidades de texto puede ser comparada determinando el coseno entre los vectores de ambas unidades. Por lo tanto, para evaluar la coherencia entre la primera y la segunda oración de un texto, el coseno entre los vectores de las dos oraciones debe ser determinado. Por ejemplo, dos oraciones que utilicen exactamente los mismos términos y con la misma frecuencia tendrán un coseno de 1; mientras que dos oraciones que usen términos que no sean semánticamente relacionados, tenderán a tener cosenos cercanos a 0, o

[2] "LSA is a fully automatic mathematical/statistical technique for extracting and inferring relations of expected contextual usage of words in passages of discourse. It is not a traditional natural language processing or artificial intelligence program; it uses no humanly constructed dictionaries, knowledge bases, semantic networks, grammars, syntactic parsers, or morphologies, or the like, and takes as its input only raw text parsed into words defined as unique character strings and separated into meaningful passages or samples such as sentences or paragraphs" (Landauer *et al.*, 1998).

menores aún, resultando un valor negativo (Foltz et al., 1998). Se añade a esto, con el fin de evitar confusiones, que si bien el valor del coseno puede fluctuar entre 1 y -1, para el caso de las predicciones de coherencia mediante Análisis Semántico Latente, tal cual señalan Foltz et al. (1998), el coseno obtenido entre los vectores de dos unidades textuales adyacentes, que contengan términos que no sean semánticamente relacionados, normalmente será un valor positivo que tienda a 0. (Todo lo expresado en el párrafo precedente se entenderá con claridad en los resultados que se presentan en el apartado 4).

De lo hasta aquí expuesto sobre la medición de coherencia mediante Análisis Semántico Latente, se resalta que es necesaria la presencia de un corpus para la construcción del espacio semántico, a la luz del cual se analizarán los textos. Este corpus debe tener dos características fundamentales: debe contener una gran cantidad de textos y estos textos deben pertenecer a un dominio temático definido. Los textos a analizar deberán pertenecer al mismo dominio temático del que provienen los que integran el corpus. Por lo mismo, la primera etapa metodológica de este trabajo, como se explicará en el apartado siguiente, tiene que ver con la recopilación del corpus.

3. Metodología de investigación

La metodología de trabajo empleada incluyó cuatro etapas. Las tres primeras no son parte del prototipo construido, sino que se enfocan en (1) la recopilación automática del corpus, (2) la preparación automática del mismo y (3) la construcción del espacio semántico. Finalmente, la última etapa es (4) la construcción del prototipo de evaluador automático de coherencia textual.

Para el presente estudio, se trabajó en un computador con sistema operativo Linux, específicamente con la distribución Ubuntu 14.04. Dicha distribución es muy sencilla de utilizar e instalar e, incluso, se puede correr en una máquina virtual dentro de un computador con Windows o Mac OS, utilizando una aplicación como Oracle VirtualBox (http://www.virtualbox.org), sin necesidad de formatear el equipo, ni abandonar el sistema operativo que se utilice normalmente.

Pese a que todo el trabajo fue realizado en Linux, en el caso del *script* desarrollado para la etapa de recopilación del corpus, se presenta el código adicional para permitir su funcionamiento en Windows. En lo relativo al *script* para preparar el corpus, al estar programado enteramente en Python, funciona sin problemas sobre Windows. Por último, la construcción del espacio semántico y el prototipo de evaluador automático de coherencia textual solo se probaron sobre Linux, aunque es posible adaptarlo a Windows como se explicará en 3.3.

3.1. Recopilación automática del corpus

Linux, en sus distintas versiones, cuenta con una consola de comandos o Terminal, cuyo intérprete de comandos por defecto –en la mayor parte de sus distribuciones– es Bash. Dentro de las aplicaciones que se pueden invocar mediante comandos de Bash se encuentra GNU Wget. Según se define en el sitio del Proyecto GNU (2014), GNU Wget es un paquete de software libre para la recuperación de archivos a través de HTTP, HTTPS y FTP, que son los protocolos de Internet más utilizados. Es una herramienta de línea de comandos no interactiva, por lo que puede ser fácilmente invocada desde *scripts*.

En otras palabras, mediante el uso de esta aplicación se pueden descargar archivos que se encuentren alojados en Internet y esto incluye a los documentos escritos en HTML (o similares) que almacenan gran parte del código que los navegadores interpretan para presentar las páginas de un sitio web. Es decir, mediante el ingreso del comando *wget* en la Terminal de Linux es posible descargar, por ejemplo, una o varias de las noticias ubicadas en una determinada dirección de Internet.

El objeto de estudio de esta investigación lo constituye el género discursivo "noticias políticas". Se eligieron noticias de este tópico en particular, por dos razones: (1) el gran volumen de material que se produce diariamente sobre el tema, lo que se vio reflejado finalmente en el tamaño del corpus; y (2) el mayor esmero que hay en la redacción de las noticias sobre este tema; si bien esto último puede considerarse algo subjetivo, en los medios de prensa se escoge con mayor cuidado a los periodistas y editores que se aboquen a este tema, por la importancia mediática del mismo.

Las noticias sobre política fueron extraídas del sitio web del diario *La Tercera*, un medio de circulación nacional en Chile. La razón para elegir este periódico es que en su sitio en Internet cuenta con un canal dedicado a las noticias sobre política, las que se encuentran ubicadas dentro del directorio localizado en http://www.latercera.com/noticia/politica. Si se ingresa esta dirección en el navegador, aparecerá un error de página no encontrada o de acceso restringido; sin embargo, todas las noticias sobre política que publica el citado medio se encuentran alojadas en este directorio y accesibles solo por algunos días. Para recopilarlas, se utilizó la aplicación *wget*. Esta permite explorar este directorio y descargar todo el contenido que haya en él al momento de la revisión. En la Figura 1 se presenta una línea de comandos similar a la utilizada en este trabajo, pero agregando algunas opciones más, con el fin de poder explicarlas para demostrar de una forma más completa las posibilidades de GNU Wget:

Figura 1: Línea de comandos de Wget

```
wget --include noticia/politica/ --wait=20 --limit-rate=20K -r -p -N -R
jpg,jpeg,gif,png,js,swf --accept shtml -U Mozilla http://www.latercera.com/
```

Lo primero es invocar el comando, simplemente, escribiendo su nombre, *wget*. A continuación, con *--include* se indica la ruta al directorio a examinar (noticia/politica/). Luego, con *--wait=20* --limit-rate=20K se ordena a la aplicación que espere 20 segundos entre el fin de la descarga de un archivo y el inicio de la descarga del siguiente; además, limita la velocidad de descarga a 20 kb/s. Esto último con el fin ético de no saturar el servidor del medio y, a la vez, protegerse de un posible bloqueo a la IP del equipo utilizado, en caso de no emplear respetuosamente el ancho de banda y los recursos del servidor al que se accede.

A continuación, el comando incluye la opción *-r* que implica que la descarga de archivos será recursiva, o sea, buscará archivos dentro de los subdirectorios que haya. La opción *-p* le indica al programa que descargue todos los archivos que son necesarios para la correcta visualización del documento HTML. La opción -N permite que si en el computador ya existe un archivo igual al que se va a descargar, se solicite al servidor la fecha de la última modificación del fichero y, únicamente,

si es más reciente del que ya se tiene, lo descargará. La opción -R permite excluir ciertos patrones (o extensiones de archivo, para el caso) que no se desea que se descarguen (en el ejemplo: jpg, jpeg, gif, png, js, swf). La opción *--accept* le indica a la aplicación el tipo de archivos que se quiere descargar, para el presente trabajo, los que tienen extensión *.shtml*. La opción -U Mozilla permite que el GNU Wget sea detectado como un navegador en particular, Mozilla Firefox, en este caso. Por último, se incluye la dirección raíz del sitio de La Tercera.

En el trabajo realizado, no se utilizaron las opciones *-p* ni *-R*, ya que solo interesaban los archivos con extensión *.shtml*. Hay que señalar que el comando escrito más arriba igual serviría para el objetivo de obtener solo los archivos citados, pero como se señaló, podrían eliminarse las opciones mencionadas, con el fin de evitar código inútil.

Hasta aquí se presenta una forma de descargar los archivos necesarios para conformar el corpus, mediante una línea de comandos de Bash. El siguiente paso es automatizar la tarea de recopilación de texto. Dicha automatización, en Linux es muy sencilla de realizar con el servicio *cron* de Linux, que sirve para automatizar la ejecución de procesos. Basta con ingresar en una Terminal de Linux el comando *crontab -e* y, tras esto, agregar una línea de código como las que se presenta en la Figura 2:

Figura 2: Ejecución automática de la línea de comandos de Wget con Cron

```
15 22 * * * nombre_de_usuario wget --include noticia/politica/ --wait=20 --limit-rate=20K -r -p -N -R jpg,jpeg,gif,png,js,swf --accept shtml -U Mozilla http://www.latercera.com/
```

La línea de código recién presentada ejecutaría la línea de comandos de Wget todos los días a las 22:15 horas. Los asteriscos equivalen a información no entregada al comando. Así, el primero de estos corresponde al día del mes, el segundo al número del mes y el tercero al día de la semana, si se quisieran especificar dichos parámetros.

3.1.1. Utilización de Wget en Windows

Para automatizar la tarea en Windows, se puede utilizar el Programador de Tareas. Previo a ello hay que poder usar *wget* en Windows. Para esto, primero hay que instalar la aplicación Cygwin que es una colección de herramientas GNU y Open Source, que proporcionan una funcionalidad similar a una distribución de Linux en ambiente Windows. La aplicación se puede descargar desde http://www.cygwin.com/, sitio en el cual también hay instrucciones para configurarla correctamente e instalar aplicaciones GNU en Cygwin, para el presente caso, Wget. Es fundamental asegurarse de que se agregue la ruta a Cygwin en la variable de entorno PATH de Windows (más información sobre cómo hacer esto se puede revisar en http://www.computerhope.com/issues/ch000549.htm).

Una vez definida la línea de comandos e instalado Cygwin, queda el paso final que es escribir un *script* en Windows para poder ejecutar desde el Programador de Tareas. Este paso, aunque parezca complejo, es bastante sencillo. Simplemente, hay

que abrir un editor de texto plano, por ejemplo el Bloc de Notas de Windows, y copiar las siguientes líneas, asumiendo que Cygwin se instaló en su ruta por defecto (C:\ cygwin\bin):

Figura 3: Código del *script* para ejecutar en Windows

```
@echo off
C:
chdir C:\cygwin\bin
bash --login -i -c 'wget --include noticia/politica/ --wait=20 --limit-rate=20K -r -p -N
-R jpg,jpeg,gif,png,js,swf --accept shtml -U Mozilla http://www.latercera.com/'
```

Nótese que entre comillas simples, al final, va la línea de comandos para operar Wget (se copió la versión más extensa que se explicó más arriba). Luego, se debe guardar este archivo asignándole un nombre a elección y asegurarse de agregarle manualmente la extensión *.bat* (no *.txt*). El archivo resultante es un archivo Batch, que es una pequeña aplicación, que al ejecutarse lleva a cabo las instrucciones que contiene. En el presente caso, conectarse al sitio de La Tercera para descargar las noticias que necesitamos para nuestro corpus.

Lo último que resta es utilizar el Programador de Tareas de Windows para configurar que el archivo Batch se ejecute automáticamente. Las instrucciones para realizar esto se pueden encontrar en la propia ayuda del sistema operativo o visitar http://windows.microsoft.com/es-xl/windows/schedule-task#1TC=windows-7. En el caso que se expone en el presente artículo, se eligió ejecutar el archivo todos los días a las 8, a las 12, a las 16, a las 20 y a las 0 horas. Mediante este método se recopiló el corpus de 7165 textos empleados en el trabajo realizado.

Con el método descrito, cada vez que el Programador de Tareas de Windows ejecute el *script* para recopilar textos, se abrirá la consola de comandos del sistema y permanecerá funcionando hasta que se complete la tarea. Obviamente, el tiempo para esto será variable: dependerá, entre otros posibles factores, de la cantidad de textos que encuentre y de la velocidad de conexión. Una complicación de lo recién expuesto, es que la aparición de la consola puede ser un elemento distractor en caso de estar trabajando en el computador o, simplemente, se corre el riesgo de que uno mismo o alguien más cierre la consola e impida la ejecución completa de la tarea. Una buena forma de prevenir este posible problema es crear otro *script* que se ejecute en segundo plano e invoque al que se guardó con la extensión *.bat*. De esta forma, la consola no aparecerá y la ejecución de la tarea será invisible para el usuario del equipo. El código para construir este nuevo *script* se presenta en la Figura 4:

Figura 4: Código del *script* para ejecutar en segundo plano

```
set objshell = createobject("wscript.shell")
objshell.run "Ruta-al-script\script.bat", vbhide
```

Al igual que el código precedente, este debe copiarse en un editor de texto plano y guardarlo con el nombre que se desee, pero con la extensión *.vbs*. En la segunda línea, luego de *objshell.run* hay que incluir la ruta completa al *script .bat*. Finalmente, hay que guardarlo y configurar el Programador de Tareas para que ejecute el *script .vbs* y no el *.bat*. Con esto, la recopilación de textos se realizará en segundo plano y no interferirá en absoluto con el usuario del equipo en los momentos de ejecución de la tarea.

3.2. Preparación automática del corpus

El funcionamiento de esta etapa se basa en un *script* programado en Python, versión 3.4.2 (https://www.python.org/). Este *script* se hace cargo de los textos recopilados y los prepara para incorporarlos al prototipo propiamente tal que permite realizar consultas sobre relación semántica y coherencia textual. Dicho *script* funciona sin problemas sobre los sistemas operativos Linux (probado en Ubuntu 14.04 y Linux Mint 16 y 17) y Windows (probado en Windows 7, 8 y 8.1).

3.2.1. Eliminación de la información innecesaria

El primer paso dentro del desarrollo fue que el *script* tomara los textos obtenidos desde La Tercera en formato *.shtml* y los transformara en un archivo de texto plano, eliminando las marcas, los caracteres especiales y toda aquella información que, si bien no corresponde a un lenguaje de marcas para la web, es considerada inútil desde el punto de vista del analizador semántico. Algunos ejemplos de esto último son el nombre del autor de la noticia, los titulares que enlazan a otras informaciones del medio, a las noticias más leídas o a las redes sociales desde las cuales se puede seguir a La Tercera, entre otros contenidos que no aportan a la meta buscada con el analizador.

Cada archivo *.shtml* con una noticia tiene una gran cantidad de información que se debe eliminar, para dejar solo el texto de esta con sus elementos de titulación. Para realizar dicha tarea, se realizaron múltiples pruebas, sobre todo con *parsers* de HTML, algunos de los cuales estaban implementados como módulos dentro de Python. Los resultados de las pruebas no fueron enteramente satisfactorios, ya que si bien estas pequeñas aplicaciones realizaban su tarea con algún grado de éxito, no siempre lograban limpiar por completo las marcas que contenían los archivos. A esto hay que agregar el hecho de que dichas aplicaciones están orientadas a eliminar las marcas de un lenguaje, pero no ayudan en absoluto en la criba de otros elementos textuales innecesarios como el nombre del autor o los enlaces a otras noticias del sitio; más aún, al eliminar las marcas del lenguaje era todavía más complicado diferenciar el texto de la noticia de estos agregados.

Por lo recién explicado, se optó por prescindir de la utilización de herramientas prediseñadas como los parsers de HTML y enfocarse en la construcción desde cero de un *script* a la medida de los archivos recopilados, que eliminara todo lo innecesario de estos y los dejara listos para su utilización. En este escenario, el lenguaje de marcas de los archivos fue algo que –paradójicamente– permitió la realización de la tarea con éxito, debido a que en la construcción del *script* se aprovechó la segmentación del documento que realizan estas marcas, para así poder determinar qué porciones debían eliminarse y cuáles debían conservarse.

El *script* primero trabaja con todos los archivos *.shtml* que haya en un directorio determinado y realiza sobre ellos operaciones recursivas con el fin de ir eliminado el contenido considerado innecesario. Un ejemplo que clarifica mucho esta etapa, es la utilización de las marcas para eliminar porciones inútiles de código y texto. Cada archivo recopilado comienza con:

<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>

Después de esta etiqueta, viene una gran cantidad de información y marcas inútiles para el objetivo del trabajo, hasta llegar a la etiqueta:

<h1 class=”titularArticulo”>

Entre una y otra, en promedio hay 47 260 caracteres de información que no sirve. Por lo tanto, en este caso, el *script* lo que realiza es simplemente borrar las etiquetas y todo lo que está incluido entre estas, con lo que se limpia gran parte del archivo. Para ello, se utilizó la siguiente línea de código en Python:

Figura 5: Código para borrar la información innecesaria de los archivos

```
1 import re #Esta línea importa el módulo de expresiones regulares de Python.
2
3 archivo_entrada = open('entrada.txt', 'r')
4 archivo_salida = open('salida.txt', 'w')
5 texto_archivo = archivo_entrada.read()
6 eliminar = re.compile('<!DOCTYPE.*?titularArticulo”>', re.I | re.S)
7 texto_archivo2 = eliminar.sub(“”, texto_archivo)
8 archivo_salida.write(texto_archivo2)
9 archivo_entrada.close()
10 archivo_salida.close()
```

Los números que están antes de cada línea no pertenecen al código, sino que indican el número de la línea para facilitar la explicación posterior. Hay que advertir que el código presentado es completamente funcional, pero se ha modificado del original con el fin de facilitar su lectura. Lo que hace el código es abrir y leer el archivo de entrada (líneas 3 y 5) y abrir el de salida para su escritura (línea 4). Luego, empleando el módulo de expresiones regulares de Python que se importó en la primera línea, busca el patrón que comienza con *‘<!DOCTYPE’ y termina con ‘titularArticulo”>’* y lo elimina al escribir el archivo de salida (líneas 6 y 7); los signos *‘.*?’* que se emplean para unir ambas porciones del patrón indican que se incluirá en el texto a borrar, lo que sea que haya entre las dos porciones del patrón, en este caso, como ya se indicó, aproximadamente 47 260 caracteres de información innecesaria. De la forma ejemplificada, se puede borrar gran cantidad de información inútil del archivo *.shtml*. Por último, se escribe el texto modificado en el archivo de salida (línea 8) y se cierran los archivos abiertos (líneas 9 y 10).

Los nombres asignados a los archivos son arbitrarios y se pueden cambiar al escribir el *script*. De hecho, en el *script* construido no se lee y se escribe un archivo cada vez que se desea ejecutar una instrucción, sino que esta tarea se realiza solo al comienzo y al final del *script*, almacenando las salidas intermedias en la memoria del equipo. Además, para abrir los archivos se empleó la sentencia *with* que no obliga a declarar que estos se cierren (por ejemplo, *"with open ('entrada', 'r') as archivo_entrada:").* Pero, como se dijo, aquí se modificó el código empleado con el fin de que sea más fácil de entender para alguien que no tenga conocimientos de Python.

3.2.2. Otras modificaciones que realiza el script

Otros reemplazos que realiza este *script* son borrar las líneas en blanco que quedan después de las operaciones aplicadas, cambiar la codificación de caracteres a UTF-8 si es que estuviera, por ejemplo, en ANSI, y, también, cambiar los caracteres especiales por los normales del español (por ejemplo, reemplaza *á* por *á*). Por ejemplo, para realizar esto último se utiliza nuevamente el módulo de expresiones regulares de Python, con el fin de buscar estos patrones y reemplazarlos por los deseados. El código empleado para ello se presenta en la Figura 6:

Figura 6: Código con los reemplazos que realiza el *script*.

```
1 import re
2
3 reemplazos = {'&aacute;':'á', '&eacute;':'é', '&iacute;':'í', '&oacute;':'ó',
'&uacute;':'ú'} #(se omitió aquí el resto de los reemplazos, por razones de espacio).
4 archivo_entrada = open('entrada.txt', 'r')
5 archivo_salida = open('salida.txt', 'w', encoding='utf8')
6 for line in archivo_entrada:
7       for src, target in reemplazos.items():
8                   line = line.replace(src, target)
9       archivo_salida.write(line)
10 archivo_entrada.close()
11 archivo_salida.close()
```

El *script* también se hace cargo de un problema particular que tienen los textos: los elementos de titulación. Normalmente, en periodismo se aceptan de tres tipos: uno imprescindible como el titular y dos optativos como el epígrafe o antetítulo (que va sobre el titular) y la bajada o subtítulo (que como su nombre indica, va debajo del titular). Si bien en los textos recopilados, al igual que en la mayoría de las noticias para internet, se prescinde del epígrafe, el problema hallado es que no había un uso uniforme de la bajada. En otras palabras, algunas noticias tenían solo titular y otras incluían titular y bajada. Por ello, este *script* se hace cargo de reconocer cuándo una noticia tiene bajada e incluye una marca antes y después de esta, con el fin de destacarla para su uso posterior, en que podría decidirse incluir la bajada en los análisis a realizar o prescindir de ella. De la misma manera, incluye una marca para el titular.

La marca de inicio del titular es *zzzinictitularzzz* y la de término es *zzztitularfinzzz*; para el caso de la bajada, las marcas son *zzzinicbajadazzz* y *zzzbajadafinzzz*. Como se puede ver, la elección de las marcas es absolutamente arbitraria.

3.2.3. Marcas de delimitación del inicio y final de cada texto

La última operación de relevancia que realiza el *script* es agregar a cada archivo procesado las marcas exigidas por la aplicación de Análisis Semántico Latente, Infomap-NLP, que permite la construcción del espacio semántico en base al corpus textual. Estas marcas son <DOC> y <TEXT> (en ese orden) antes del inicio de cada texto, y </TEXT> y </DOC> al final de cada una de las noticias. El código para realizar dicha tarea se muestra en la Figura 7:

Figura 7: Código que agrega a los textos las marcas que requiere Infomap-NLP

```
1 archivo_entrada = open('entrada.txt', 'r')
2 archivo_salida = open('salida.txt', 'w')
3 inicio = ('<DOC>\n<TEXT>\n') + archivo_entrada.read()
4 final = inicio + ('\n</TEXT>\n</DOC>\n')
5 archivo_salida.write(final)
6 archivo_entrada.close()
7 archivo_salida.close()
```

Como una muestra de la potencia y versatilidad de Python, se expone a continuación una sola línea de código que realiza las mismas tareas que las líneas 1 a 5 del código recién presentado, aunque es un poco más complicada de leer:

Figura 8: Ejemplo del código de la Figura 7 en una sola línea

```
open('salida.txt', 'w').write('<DOC>\n<TEXT>\n' + open('entrada.txt').read()+'\n</
TEXT>\n</DOC>\n')
```

Finalmente, el *script* une todos los archivos resultantes en un solo fichero de texto plano. Esto se realiza gracias a los módulos *glob*, *shutil* y *os* de Python. En el ejemplo que sigue, se asume que los archivos a unir están ubicados en el directorio llamado *textos*, que se encuentra junto al script:

Figura 9: Código que une todos los archivos en uno solo

```
1 import glob,shutil,os
2
3 ruta = os.getcwd() + '/textos/'
4 with open('salida_marcas.txt', 'wb') as archivo_salida:
```

```
5    for filename in glob.glob(os.path.join(ruta, '*.txt')):
6        with open(filename, 'rb') as archivo_entrada:
7            shutil.copyfileobj(archivo_entrada, archivo_salida)
```

3.2.4. Función para elegir el formato de las noticias en el archivo de salida

Lo hasta aquí presentado es la forma de procesamiento de datos y metodología de funcionamiento del *script.* Obviamente, se pueden realizar muchas tareas adicionales escribiendo las líneas de código para ello, con el fin de adaptarlo a las necesidades puntuales de cada situación. Por ejemplo, en el caso específico del *script* construido se definió una función que permite elegir si el archivo de salida lo queremos con o sin las marcas (*zzzinictitularzzz, zzztitularfinzzz*, *zzzinicbajadazzz, zzzbajadafinzzz*) y con o sin la bajada del texto. Dicha función nos da a elegir, mediante una pregunta que se imprime en la consola de comandos, si queremos el archivo completo y con marcas (opción 1), sin marcas y con bajada (opción 2), o sin marcas y sin bajada (opción 3). Esto se logra gracias al uso del condicional *if* en el código, como se muestra en la Figura 10, en que se presenta la función escrita (llamada *elegir*), que trabaja sobre el archivo *salida_marcas.txt* generado en los pasos descritos más arriba:

Figura 10: Código de la función que permite elegir si se incluyen las marcas en el archivo de salida

```
def elegir(pregunta, reclamo='Se equivocó de tecla'):
    while True:
        ok = input(pregunta)
        if ok in ('1'):
            return True
        if ok in ('2'):
            with open('salida_marcas.txt') as archivo_entrada:
                paso1 = filer.read()
                archivo_salida = open('salida_conbajada.txt', 'w')
                paso2 = re.sub('zzzinictitularzzz','', paso1)
                paso3 = re.sub('zzztitularfinzzz','', paso2)
                paso4 = re.sub('zzzinicbajadazzz','', paso3)
                paso5 = re.sub('zzzbajadafinzzz','', paso4)
                archivo_salida.write(paso5)
                archivo_salida.close()
            os.remove('salida_marcas.txt')
            return True
        if ok in ('3'):
            with open('salida_marcas.txt') as archivo_entrada:
                paso6 = filer.read()
                archivo_salida = open('salida_sinbajada.txt', 'w')
                patter = re.compile('zzzinicbajadazzz.*?zzzbajadafinzzz\n', re.I | re.S)
                paso7 = patter.sub("", paso6)
```

```
            paso8 = re.sub('zzzinictitularzzz','', paso7)
            paso9 = re.sub('zzztitularfinzzz','', paso8)
            archivo_salida.write(paso9)
            archivo_salida.close()
        os.remove('salida_marcas.txt')
        return True
    print(reclamo)

elegir('¿Qué archivo quiere: completo con marcas(1), completo sin marcas(2), sin marcas y sin bajada(3)? =')
```

Por último, es importante señalar que la ejecución del *script*, si es que se procesan muchos archivos de texto a la vez (los 7165, por ejemplo), demanda una gran cantidad de recursos del computador, ya que el procesamiento de cantidades ingentes de cadenas de texto es una tarea compleja para cualquier equipo. Lo anterior, junto con las especificaciones de la máquina en que se ejecute el *script*, influirá en el tiempo que tarde en completarse la tarea.

3.3. Construcción del espacio semántico

La creación del espacio semántico se realiza a través de un *script* que se encarga únicamente de tomar el archivo con los textos entregados por la etapa de preparación de los mismos y, en base a este fichero, previa lematización de los textos que lo conforman, construye el espacio semántico multidimensional con el que se trabajará. Para ello se utiliza Python y NLTK (Natural Language Toolkit, disponible en http://www.nltk.org/).

La tarea de construir este espacio semántico en Infomap-NLP, sin considerar la lematización, es bastante sencilla y se puede realizar rápidamente mediante un sencillo *script* de texto que ejecute las tres líneas siguientes:

Figura 11: Código para construir el espacio semántico

```
INFOMAP_WORKING_DIR=/ ruta_al_directorio_de_trabajo_elegido
                                                    (fija el directorio de trabajo)
export INFOMAP_WORKING_DIR
                                                    (exporta el directorio)
infomap-build -s /ruta_al_archivo_con_el_corpus  nombre_elegido_para_el_espacio
                                                    (construye el espacio)
```

El *script* se basó en las instrucciones para realizar esta tarea en forma manual, entregadas en el sitio de Infomap-NLP (http://infomap-nlp.sourceforge.net/doc/user_manual.html), y se utilizó la construcción del modelo en un directorio de trabajo y no en un directorio permanente, porque de esta forma se tiene mayor control sobre el mismo. La única gran repercusión de esto es la forma de realizar las posteriores

consultas, para lo cual solo hay que indicar la ruta al directorio del modelo y utilizar el comando *associate* de Infomap-NLP. La forma de realizar las consultas al modelo en forma manual también se indican en la URL recién mencionada, pero es básicamente la estructura que sigue:

associate -t -c espacio_modelo presidente

En el ejemplo recién expuesto, lo que se está realizando es consultar cuáles son las palabras que tienen una mayor relación semántica con *presidente*, en el espacio semántico denominado como *espacio_modelo* (el nombre asignado es arbitrario). Dicha consulta se puede realizar inmediatamente tras crear el espacio, ya que el Análisis Semántico Latente no precisa de entrenamiento para operar.

Por otro lado, la parte de invocar al lematizador que incluye este *script*, es exactamente igual a como opera en el prototipo de evaluación automática de coherencia textual, por lo tanto se detallará únicamente para este último en el apartado siguiente.

A diferencia de los *scripts* de 3.1. y 3.2., este necesita trabajar en un ambiente Linux, ya que la aplicación Infomap-NLP funciona nativamente en un sistema operativo de este tipo. Sin embargo, debería ser factible operarlo en un sistema con Windows, si es que se instalara Infomap-NLP utilizando la aplicación Cygwin, ya que según las notas de lanzamiento de la última versión de Infomap-NLP, es posible de realizar. Sin embargo, para efectos del presente trabajo, esto no ha sido testeado. Obviamente, las instrucciones sobre cómo instalar Infomap-NLP no se incluyen, pero pueden revisarse en http://infomap-nlp.sourceforge.net/doc/user_manual.html. En el caso del presente trabajo, para facilitar la tarea, se creó una máquina virtual con la aplicación Oracle VirtualBox (https://www.virtualbox.org/), en la cual se utilizó el sistema operativo Ubuntu 14.04, sobre el que se instaló Infomap-NLP. VirtualBox se instaló sobre un computador con Windows 8.1, por lo que no hubo necesidad de formatear el PC ni resolver ninguna otra complicación adicional. De hecho, solo se utilizaron 30GB de disco duro en el equipo con Windows y la instalación completa no tomó más de 45 minutos.

3.4. Evaluador automático de coherencia textual

Para el desarrollo del prototipo de evaluador automático de coherencia textual, al igual que para el *script* que construye el espacio semántico, se utilizó Python y NLTK. Además, al igual que el *script* explicado en 3.3., utilizar Infomap-NLP requiere trabajar en un ambiente Linux. Si bien su funcionamiento es más complejo que el de los tres *scripts* anteriores, expresándolo en términos simples, se encarga de las diferentes tareas que se necesitan para realizar la comparación de las unidades textuales de cada noticia procesada en el prototipo.

3.4.1. Lematización de los textos

Como se mencionó antes, tanto en el *script* de 3.3. como en el presente, se requiere realizar un proceso de lematización. Para ello, se empleó el Snowball Stemmer, incluido en NLTK, ya que trabaja con lengua española y, además, con un grado de precisión bastante aceptable. En ambos casos, para poder utilizar Snowball, los textos

deben ser preparados previamente, eliminando los caracteres que no son admitidos (tildes, comas, signos de interrogación, signos de exclamación, etc.). Para realizar esto, se utilizó el módulo de expresiones regulares de Python, tal cual se explicó en 3.2. En el caso de la lematización, el código que se empleó se presenta en la Figura 12:

Figura 12: Código utilizado para lematizar los textos

```
from nltk.stem.snowball import SnowballStemmer #Esta línea
importa el lematizador desde NLTK.

archivo_entrada = open('entrada.txt', 'r')
archivo_salida = open('salida.txt', 'w')
for linea in archivo_entrada:
   linea1 = re.sub('\n', ' <zzz>\n', linea)
   singles = []
   stemmer = SnowballStemmer("spanish")
   for plural in linea1.split():
      singles.append(stemmer.stem(plural))
      linea2 = (' '.join(singles)).replace('<zzz>','\n').lstrip()
   archivo_salida.write(linea2)
archivo_entrada.close()
archivo_salida.close()
```

3.4.2. Comparación de unidades textuales adyacentes

En el caso del *script* construido, no se busca averiguar las palabras que mayor relación semántica tienen con una palabra en particular, sino que el objetivo es comparar todas las unidades textuales adyacentes –párrafos para el caso–, de una noticia en particular. El código que se presenta en la Figura 13 es el utilizado para realizar esta tarea, el que debería agregarse a continuación del código para lematizar ya presentado (se reproduce la línea a partir de la cual debería añadirse):

Figura 13: Código que compara las unidades textuales adyacentes

```
import subprocess

modulo_salida = []
for linea in entrada:
#—Código omitido—#
      linea2 = (' '.join(singles)).replace('<zzz>','\n').lstrip()
      linea_texto = line2.split()
       comandos_infomap = ['associate', '-q', '-m', '/home/nombre/
espacio2', '-c', 'espacio_modelo']
      procesar = comandos_infomap + linea_texto
```

```
        vectores = subprocess.check_output(procesar)
        modulo_salida .append(vectores)
    modulo_salida  = [x for x in salidas if x != b' ']
```

El prototipo, en esta etapa, toma cada uno de los textos del módulo de entrada por separado, los segmenta en párrafos y calcula el vector que representa a cada uno de estos párrafos. Luego, entrega los valores de cada vector, con el fin de poder pasar a la última tarea que realiza este *script* del prototipo: la comparación de todas las unidades textuales adyacentes, a través del cálculo del coseno del ángulo que forman los vectores que representan a cada unidad. Este coseno es igual al producto escalar entre los vectores, dividido por el producto de sus módulos. Se puede representar como sigue:

$$\cos(A,B) = \frac{\sum_{i=1}^{n} a_i b_i}{\|A\| * \|B\|}$$

Las operaciones para realizar dicho cálculo se realizan con la ayuda del módulo Numpy de Python (http://www.numpy.org/). El código utilizado en el *script* se presenta en la Figura 14 y también debería ser agregado luego del código que calcula los vectores:

Figura 14: Código que compara el valor de los vectores de las unidades textuales adyacentes

```
import numpy as np
from numpy import linalg as LA

modulo_salida  = []
for linea in entrada:
#—Código omitido—#
   modulo_salida  = [x for x in salidas if x != b' ']
   valores =[]
   for par1, par2 in zip(modulo_salida[0::1], modulo_salida[1::1]):
      a = np.genfromtxt([par1])
      b = np.genfromtxt([par2])
      coherencia = (np.dot(a, b))/(LA.norm(a)*LA.norm(b))
      valores.append(coherencia)
promedio = np.mean(valores)
```

El código de la Figura 14 toma el valor de los vectores que representan a cada una de las unidades textuales definidas (párrafos) y compara el primer valor con el segundo, el segundo con el tercero y así sucesivamente, hasta llegar a la última pareja. Los puntajes de coherencia obtenidos al comparar dos unidades textuales se promedian y este resultado es el puntaje de coherencia del texto en su conjunto.

Una vez realizadas todas estas operaciones, el *módulo de salida* arroja los resultados en un archivo de texto plano que permite revisarlos y someterlos a algún tipo de procesamiento posterior, como se requerirá en las etapas siguientes del trabajo. A continuación, se muestra un ejemplo de la salida final en que se copia el titular de la noticia, los puntajes de coherencia para cada pareja de vectores y, por último, el puntaje del texto como unidad.

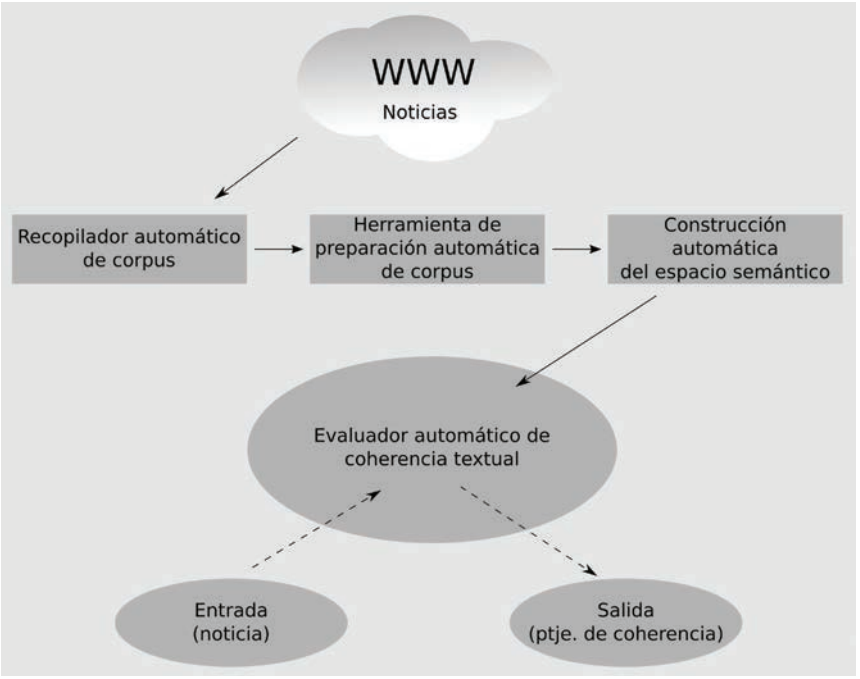
Walker: “Esto es un tema político más que disciplinario”

0.66224260856399708
0.73623747060453659
0.79895212424025353
0.67184566234764576
0.52964567682386254
0.605330035033259

0.667375596269

El funcionamiento del prototipo desarrollado y de los tres *scripts* de las etapas previas (recopilación y preparación del corpus, construcción del espacio semántico) puede resumirse gráficamente en la Figura 15:

Figura 15: Esquema integrado de funcionamiento del prototipo y de los tres *scripts* de las etapas previas



4. Resultados de las pruebas realizadas al prototipo

La forma de comprobar si la máquina evalúa acertadamente la coherencia textual de una noticia se efectúa comparando los resultados que esta entrega con el juicio de evaluadores humanos. En Hernández y Ferreira (2010) se utilizó y validó un método de análisis similar al propuesto en el presente trabajo, por lo que no fue necesario replicar toda la tarea de comparación de los resultados.

El prototipo se sometió a una serie de pruebas en que se alteraba intencionadamente la coherencia textual con el fin de ver si era sensible a estos cambios: manipular la coherencia de textos, crear textos sin sentido uniendo párrafos extraídos desde diferentes noticias (cuya única relación era pertenecer al dominio en que se enfocó el trabajo) y comparar textos que presentaban la misma noticia, pero extraída de medios de prensa diferentes (para ello se trató a cada noticia como una única unidad textual).

Entre las pruebas realizadas se compararon oraciones construidas, alusivas a temas pertenecientes al dominio. Por ejemplo, al cotejar la secuencia *"el diputado votó en contra de la ley"* con *"el senador rechazó el proyecto"* el puntaje de coherencia que arroja el prototipo es de 0.81, dicha cifra refleja la similitud semántica de ambas oraciones y también da cuenta de los matices que la diferencian. Si se cambia la segunda oración por *"el ministro fue a ver a la presidenta"* el puntaje de la comparación desciende a 0.13, dando cuenta de la poca relación entre ambas secuencias.

Otra prueba consistió en tomar noticias con un puntaje de coherencia relativamente alto (superior a 0.60) y manipular el texto cambiando palabras, con el fin de alterar la coherencia textual. Por ejemplo, a continuación se presenta el titular y el primer párrafo de una noticia en particular que tiene un puntaje promedio de 0.69 (considerando el texto completo):

> **Titular de comisión de RR.EE. del Senado y dichos de Sabag: "No representan el espíritu de Chile"**
> El presidente de la comisión de RR.EE. del Senado, Francisco Chahuán (RN), desestimó la polémica que generaron en Bolivia los dichos de su par de la Cámara de Diputados, Jorge Sabag (DC), quien sostuvo en una entrevista con el diario El Sur que "a Chile le ha ido mejor con armas que con diplomacia", en relación al inicio de los alegatos del juicio entre ambos países en la Corte Internacional de Justicia de La Haya.

El puntaje que arroja la comparación entre el titular y el primer párrafo es de 0.71. Sin embargo, al cambiar algunas palabras en el primer párrafo el puntaje desciende a 0.45, por lo que el prototipo demuestra ser sensible a los cambios que alteran la coherencia. La misma tendencia se observó en las otras nueve pruebas realizadas. A continuación se presenta el párrafo modificado, destacando en cursivas las palabras cambiadas.

> El *intendente* de la comisión de *vivienda* del *ministerio*, Francisco Chahuán (RN), desestimó la polémica que generaron en Bolivia los dichos de su par de la *Moneda*, *Fuad Chaín* (DC), quien sostuvo en una entrevista con el diario El Sur que "a *Concepción* le ha ido mejor con armas que con diplomacia", en relación al inicio de los alegatos del juicio entre ambos países en la Corte Internacional de Justicia de La Haya.

Otra prueba realizada consistió en tomar cinco textos que tuvieran un puntaje superior a 0.60 y construir un único texto combinando el titular de un texto, el primer

párrafo de otro, el segundo párrafo del siguiente y así sucesivamente. Para realizar este constructo, se decidió que los cinco textos originales debían pertenecer al mismo medio. A continuación se presenta el texto resultante de la combinación de cinco noticias de Bío Bío Chile.

> **Diputado que pidió minuto de silencio por Pinochet: "Le debemos un eterno agradecimiento"**
> Alcaldes de la UDI acordaron levantar la candidatura del jefe comunal de Las Condes, Francisco de La Maza, a la presidencia de la colectividad. Mario Olavarría, afirmó que ni el senador Hernán Larraín, ni el diputado Javier Macaya son los que el gremialismo necesita. "Yo espero que algún día tengamos un gobierno realmente de derecha en nuestro país y ahí podamos reescribir la historia realmente, como corresponde, y no como se ha escrito todos estos años", agregó.
> En la sesión de hoy, concurrió nuevamente el ex jefe de la División de Administración y Finanzas, Gabriel Aldana, dado que los parlamentarios querían aclarar sus afirmaciones a la luz de otras versiones ante la Comisión, emitidas con posterioridad a su primer testimonio. La mantención de la cautelar fue valorada por el abogado de Carlos Lavín y Carlos Délano, Julián López. "El hecho de estar sometido a una medida cautelar de menor intensidad facilita el trabajo de la defensa, que nos va a permitir demostrar que la participación que se les atribuye a mis representados no es la que la fiscalía y los acusados han sostenido hasta ahora", sostuvo.

Como era esperable, para el texto recién presentado el puntaje fue de 0.26. Lo mismo sucedió en los otros nueve textos construidos con puntajes que fluctuaron entre 0.21 a 0.34. Una última prueba realizada es la comparación de la misma noticia publicada en diferentes medios. Sin duda, esta es una de las experiencias más interesantes porque compara textos de diferentes fuentes, pero referidos a un mismo hecho. Para ello, se utilizaron textos de *La Tercera* (www.latercera.com), *Emol* (www.emol.com), *Cooperativa* (www.cooperativa.cl) y *Bío Bío Chile* (www.biobiochile.cl).

A continuación se detallan las cinco pruebas realizadas para el caso de una de esas noticias en particular, para, finalmente, presentar una tabla y un gráfico que resumen todos los resultados que arrojaron las diez noticias a que se aplicaron estas pruebas.

El texto íntegro de la noticia en que se enfocó el ejemplo, titulada "Mariana Aylwin entrega descargos a tribunal supremo DC y dice que solicitud de expulsión es injusta" (*La Tercera*), se comparó con el texto de "Mariana Aylwin y petición de expulsión: Vengo a defenderme de una acusación infundada e injusta" (*Emol*). Los resultados de dicha comparación entre dos textos diferentes, pero enfocados exactamente en el mismo tema, arrojaron un puntaje de 0.94, que indica una altísima relación semántica entre ambos escritos, lo que demuestra la efectividad de la medición realizada.

En una segunda prueba, se mantuvo el texto publicado por *La Tercera*, pero ahora se le comparó con otra noticia relacionada que publica Emol, sobre la opinión del presidente del partido político aludido en las noticias anteriores, titulada "Presidente de la DC: Más que un tema disciplinario es político". El resultado, obviamente, disminuye a 0.78. Dicho puntaje también expresa una alta relación semántica entre los textos, ya que tratan sobre el mismo tema, pero la baja en el puntaje se explica porque no son la misma noticia, sino que solo son textos relacionados.

En una tercera prueba, se comparó la misma noticia de *La Tercera* con un texto sobre política de Emol, pero que no tenía relación con el fondo de la noticia original.

Este último se titula "Rossi abandona indignado comisión que analiza indicaciones de la reforma educacional". En esta comparación, el puntaje disminuyó aún más, al arrojar sólo 0.54, lo que da a entender que se trata de noticias diferentes, aunque pertenecientes a un mismo dominio temático (política).

En una cuarta prueba se comparó el mismo texto utilizado en los testeos ya mencionados, con una noticia ajena al dominio temático elegido, titulada "Quintanilla repite su tercer lugar y avanza a la cuarta posición de la General del Dakar en motos". En esta ocasión, el puntaje entregado por el prototipo fue de 0.36, que es bastante bajo.

Por último, en una quinta prueba se comparó el texto original con una noticia de un dominio temático diverso, pero además externa a Chile, país en el que se produjeron todas las noticias utilizadas en las cuatro primeras pruebas. La nota se titula "Actor de Star Wars y James Bond muere durante ensayo de obra de teatro" y en la comparación se obtuvo el más bajo puntaje de todas las pruebas con solo 0.32.

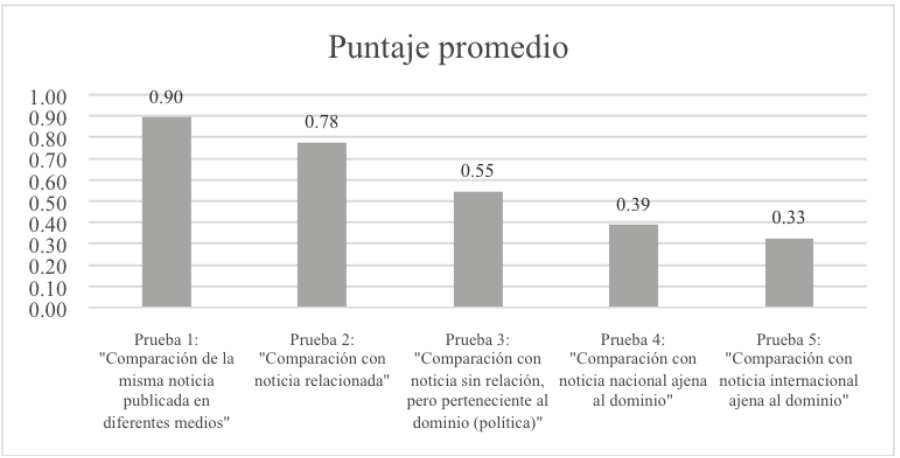
En lo referente a la comparación de textos completos, se realizaron las cinco pruebas descritas a diez noticias en total, como ya se señaló, y en todos los casos, las bajas de puntaje presentaron tendencias similares. En seis de las pruebas se utilizó como texto base uno de *La Tercera*, que es el medio del que se extrajeron las noticias con que se construyó el corpus empleado para crear el espacio semántico, y se las comparó con cinco noticias extraídas de otros medios: *Emol*, *Cooperativa* y *Bío Bío Chile*. En dos pruebas se utilizó una noticia de *Emol* como base y se la comparó, en cada caso, con cinco noticias provenientes de *La Tercera*. Por último, se realizaron dos pruebas utilizando como base una noticia de *Cooperativa* y otra de *Bío Bío Chile*, con el fin de realizar la experiencia en textos publicados en sitios informativos que no tienen su raigambre en la prensa escrita como *Emol* y *La Tercera*, sino que en radios. La idea de la experiencia era utilizar en su mayoría textos que no fueran de *La Tercera*, por ello, en total –incluyendo textos base y de comparación– se emplearon 16 textos extraídos del citado medio y 44 de otros: 12 de *Emol*, 16 de *Cooperativa* y 16 de *Bío Bío Chile*. A lo señalado, hay que añadir que la selección de los textos base se realizó al azar de entre noticias recopiladas desde el sitio de *La Tercera* a lo largo de seis meses y que no forman parte del corpus integrado en el prototipo. El universo total para esta selección fue de 2510 textos. Se eligieron diez, y de esos, seis se utilizaron como texto base (los de *La Tercera*); para los cuatro casos restantes, se tomó el tema de la noticia de *La Tercera* y se buscó en los otros medios ya señalados la noticia equivalente. En la Tabla 1 se presentan los resultados de toda la experiencia recién descrita.

Como se puede ver en la Tabla 1, la tendencia a la baja de los puntajes es similar en todos los casos. La única vez en que esto no se cumple es en la prueba 4 de la segunda noticia, ya que el prototipo arrojó que la relación semántica entre la noticia base y una noticia nacional ajena al dominio fue de 0.41, en circunstancias que la comparación con la noticia internacional ajena al dominio fue de 0.45. Se considera como no relevante este caso, sobre todo por la tendencia a no haber una diferenciación significativa entre las comparaciones de las pruebas 4 y 5, en donde los puntajes tienen una separación cuantitativa mucho menor que el resto de las pruebas. Por último, en el Gráfico 1 se muestra el promedio de los puntajes obtenidos en las cinco pruebas realizadas a las diez noticias, con el fin de ver cómo varía la medición de una forma general, a medida que se reduce la relación semántica entre los textos comparados.

Tabla 1: Resultados de la comparación de noticias completas pertenecientes a distintos medios

Título noticia base	**Medio texto base**	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**	**Medio textos comparados**
Mariana Aylwin entrega descargos a tribunal supremo DC y dice que solicitud de expulsión es injusta	La Tercera	0,94	0,78	0,54	0,37	0,32	Emol
Fulvio Rossi por dichos de rector Sánchez sobre aborto: "Aquí nadie está por sobre la ley, y él tampoco"	La Tercera	0,86	0,75	0,55	0,41	0,45	Emol
De la Maza renuncia a directiva UDI y se agudiza crisis interna	La Tercera	0,89	0,84	0,60	0,42	0,32	Bío Bío
Diputado que pidió minuto de silencio por Pinochet: "Cómo se le ocurre que voy a estar arrepentido"	La Tercera	0,92	0,78	0,56	0,37	0,30	Bío Bío
Velasco reaparece junto a Amplitud y profundiza distancias con Nueva Mayoría	La Tercera	0,90	0,79	0,51	0,40	0,28	Cooperativa
Pizarro y Chahín fijan plazo de 48 horas para explorar lista unitaria	La Tercera	0,87	0,76	0,59	0,31	0,23	Cooperativa
Bachelet cita a comité político extraordinario para dar instrucciones sobre agenda anticorrupción	Emol	0,83	0,65	0,51	0,34	0,30	La Tercera
SQM: Contadora admitió entrega irregular de dinero a Martelli para campaña presidencial de Frei	Emol	0,94	0,78	0,52	0,40	0,35	La Tercera
Corte mantiene en arresto domiciliario a ex controladores de Penta	Bío Bío	0,88	0,87	0,54	0,45	0,30	Cooperativa
Ministra Ximena Rincón alertó que su correo electrónico fue hackeado	Cooperativa	0,91	0,78	0,56	0,44	0,43	Bío Bío

Gráfico 1: Puntajes promedio de todas las pruebas



5. Conclusiones y proyecciones

Como se dijo al inicio del artículo, dos fueron los objetivos del presente trabajo: contribuir en la investigación sobre el procesamiento semántico automático en nuestra lengua y diseñar e implementar una herramienta que evalúe la coherencia textual de noticias, en forma automática y eficiente. La construcción de este módulo de procesamiento semántico, como se señaló, se enmarca en la tarea mayor de diseñar un sistema tutorial inteligente destinado a apoyar la producción escrita de noticias en estudiantes de periodismo. La importancia de este trabajo radica en que la noticia es un texto complejo desde el punto de vista de su elaboración para los estudiantes de periodismo: exige que el autor comprenda el hecho sobre el que escribe y tenga la capacidad de resumirlo sin restarle nada de la información fundamental que involucra.

Pese al objetivo particular con que se desarrolló la herramienta descrita en el presente artículo, un prototipo como el diseñado, al basarse en *scripts* de texto, permite su adaptación a múltiples proyectos en que se requiera realizar análisis semántico. De hecho, bastaría con cambiar el corpus a partir del cual se construye el espacio semántico y adaptar algunas líneas de código para tener una herramienta personalizada para trabajar en otro ámbito que se desee.

De las pruebas realizadas en el testeo del prototipo se puede colegir que no afecta a su correcto funcionamiento, el que las noticias analizadas provengan de un medio de prensa distinto al utilizado para recopilar las noticias que conforman el corpus, en base al cual se construyó el espacio semántico multidimensional: La Tercera, en este caso. Si bien esto era algo esperable, ya que en un medio de prensa no es un solo periodista quien escribe todos los textos, sino que un equipo de redacción, era importante despejar la duda mediante pruebas empíricas. Lo mencionado es algo fundamental para el trabajo posterior de integrar el prototipo en el sistema tutorial inteligente al que se alude más arriba, ya que permite procesar textos escritos por cualquier autor y, en este caso en particular, estudiantes de periodismo.

Es importante, además, señalar que en el testeo realizado hubo que corregir manualmente la ortografía de los textos utilizados, ya que los errores de este tipo alteraban los puntajes arrojados por el prototipo. Por lo mismo, se determinó que una etapa necesaria en el futuro de este trabajo es la construcción de un corrector ortográfico que pueda ser incorporado al prototipo y al sistema tutorial inteligente; más aún, cuando los textos a analizar serán de estudiantes de pregrado.

Para finalizar, es necesario señalar que la herramienta diseñada, como parte de un proyecto mayor que se enfoca en que estudiantes de periodismo aprendan y practiquen la escritura de noticias, necesita incorporar otros dominios temáticos al ya existente de política, con el fin de que el producto final sea lo más completo posible y pueda ser utilizado como un complemento a la enseñanza en el aula.

Obras citadas

Álvarez, Gerardo. 1995. *Textos y discursos*. 2ª ed. Concepción, Chile: Universidad de Concepción. 12, 13, 97-122.

Boonthum, C., Levinstein, I., & McNamara, D.S. 2007. "Evaluating self-explanations in iSTART: Word matching, latent semantic analysis, and topic models". In A. Kao & S. Poteet (Eds.), *Natural Language Processing and Text Mining*. Pp. 91-106. Londres: Springer-Verlag UK.

De Beaugrande, R. & Dressler, W.U. 1997. *Introducción a la lingüística del texto*. Barcelona: Ariel.

De Vega, Manuel et al. 1999. "Procesamiento del discurso". *Psicolingüística del español*. Comp(s). Cuetos, Fernando y De Vega, Manuel. Pp 273. España: Trotta.

Ferreira, A. (2004). *Feedback Strategies for Second Language Teaching with Implications for Intelligent Tutorial Systems*. Scotland:University of Edinburgh UK.

Ferreira, A., Salcedo, P., Kotz, G. y Barrientos, F. (2012). "La Arquitectura de ELE-TUTOR: Un Sistema Tutorial Inteligente para el Español como Lengua Extranjera". *Revista Signos*, 45(79), 102–131. [En línea]. Disponible en: http://www.scielo.cl/pdf/signos/v45n79/a01.pdf [Consulta: 27-7-2015].

Foltz, P., Kinstch, W. y Landauer, T. 1998. "The measurement of textual coherence with Latent Semantic Analysis". Disponible en: http://lsa.colorado.edu/papers/dp2.foltz.pdf [Consulta: 20-10-2014].

GNU Project. 2014. [En línea]. Disponible en: https://www.gnu.org/software/wget/ [Consulta: 17-10-2014].

Hernández, Sergio y Ferreira, Anita. 2010. "Evaluación automática de coherencia textual en noticias policiales utilizando Análisis Semántico Latente". *RLA*. Revista de Lingüística Teórica y Aplicada Concepción (Chile), 48 (2), II Sem. 2010, pp. 115-139.

Halliday, M.A.K. & Hasan, R. 1976. *Cohesion in English*. Londres: Longman.

Jurafsky, Daniel y Martin, James. 2008. *Speech and language processing*, 2ª edición. Pp. 685. Usa: Prentice Hall.

Landauer, T. 2002. "On the computational basis of learning and cognition: Arguments from LSA". [En línea]. Disponible en: http://lsa.colorado.edu/papers/Ross-final-submit.pdf [Consulta: 07-09-2014].

Landauer, T.; Foltz, P. y Laham, D. 1998. "An Introduction to Latent Semantic Analysis. Discourse Processes". *Discourse Processes* 25(2&3), 259-284. [En línea]. Disponible en: http://lsa.colorado.edu/papers/dp1.LSAintro.pdf [Consulta: 07-09-2014].

Louwerse, Max. 2004. "Un modelo conciso de cohesión en el texto y coherencia en la comprensión". *Revista Signos*, vol.37, n.56. Pp. 41-58. [En línea]. Disponible en: http://www.

scielo.cl/scielo.php?pid=S0718-09342004005600004&script=sci_arttext [Consulta: 20-05-2016].

Martín Vivaldi, Gonzalo. 1993. *Géneros periodísticos*. Pp 369. Madrid, España: Editorial Paraninfo.

Martínez Albertos, José Luis. 2004. *Curso general de redacción periodística*, 5ª edición, 3ª reimpresión. Pp 288. Madrid, España: Editorial Paraninfo.

McCarthy, P., Briner, S., Rus, V. y McNamara, D. 2007. "Textual signatures: Identifying text-types using latent semantic analysis to measure the cohesion of text structures". *Natural language processing and text mining*. Comp(s). A. Kao y S. Poteet. Londres: Springer-Verlag UK. 107-122.

Venegas, René. 2003. "Análisis Semántico Latente: una panorámica de su desarrollo". *Revista Signos* 36:121-138.