Sistema de apoyo lingüístico en español para personas sordas

Obdulia Pichardo-Lagunas, Beatriz Torres-Alatriste, Bella-Citlali Martínez-Seis, Víctor-Darío Cuervo-Pinto, Miguel-Alejandro Martínez-Rosales

Instituto Politécnico Nacional,
Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas, D.F., México

therasmus_bet@hotmail.com,
{opichardola, vdcuervo, bmartinezs,mamartinezr}@ipn.mx

Resumen. Se propone un sistema computacional que sirve como herramienta de apoyo a personas sordas en el aprendizaje de la lengua española escrita. La aplicación desarrollada utiliza herramientas de Procesamiento de Lenguaje Natural para el análisis y validación de los resultados obtenidos en los problemas propuestos. Los dos ejercicios planteados en el sistema tienen como fin desarrollar habilidades de escritura de la lengua española en las mencionadas personas. En el primer caso se implementa la validación sintáctica en la construcción de oraciones con unidades léxicas previamente definidas y en el segundo el análisis léxico y sintáctico en la escritura de oraciones relacionadas a tópicos específicos planteados por especialistas en la enseñanza a personas sordas.

Palabras clave: Sordo, procesamiento de lenguaje natural, lingüística computacional, sistema computacional, analizador sintáctico, detección de errores.

System for Linguistic Support in Spanish for Deaf Persons

Abstract. We propose a computational system to be used by deaf people as a tool in their written Spanish language learning. Natural Language Processing techniques were used to build the computer program in order to analyze and validate obtained results in the offered exercises. Two problems are offered in the computer program. Both of them are intended to improve Spanish writing abilities in deaf people. A syntactic validation is carried out in the first proposed exercise. This is done when the user builds sentences in written Spanish. Those problems use a previously defined lexicon. In the second kind of problems a lexical and syntactic analysis is performed. The lexicon and the syntax are related to specific topics in Spanish teaching for deaf people.

Keywords: Deaf, natural language processing, computational linguistics, computer system, parser, error detection.

1. Introducción

El presente trabajo se centra en el desarrollo de una herramienta para el uso de las personas hipoacústicas. En el año 2010, las personas que tuvieron algún tipo de discapacidad fueron 5 739 270 en México. Las personas con limitación para escuchar fueron un número de 694 464 [1]. Por los motivos anteriores se considera pertinente el desarrollo de herramientas para su apoyo.

a. Problemática

Las lenguas humanas constituyen entes de gran magnitud. Por ello requieren, en ocasiones, acercamientos de orden estadístico para su estudio [2]. Se decidió enfrentar el problema de la magnitud de la lengua mediante el uso de una sintaxis y un léxico reducidos.

Un lenguaje formal es un conjunto de cadenas de símbolos sobre un alfabeto, al que se denomina vocabulario. Los lenguajes que usan las personas para su comunicación interpersonal son llamados lenguajes naturales. Una de las diferencias entre los lenguajes naturales y los formales es la complejidad de sus estructuras de conocimiento. El tratamiento de esta complejidad requiere del uso de formalismos específicos para la representación de conocimiento lingüístico, así como algoritmos y técnicas computacionales adecuadas [3].

El problema a resolver en este artículo resulta en desarrollar un sistema que proporcione funciones de análisis léxico y sintáctico de oraciones en español para su uso por personas sordas. Los usuarios deben conocer el abecedario, algunas palabras y algunas estructuras gramaticales del español.

b. Marco teórico

El Procesamiento de Lenguaje Natural (PLN) es el reconocimiento y utilización de la información expresada en un lenguaje humano a través de sistemas informáticos [4]. El PLN comprende diferentes niveles de análisis, a saber el morfológico, el sintáctico, el semántico y el pragmático [5]. Los formalismos lógicos permiten dichos análisis. Dentro de los diferentes problemas que se presentan en el procesamiento del lenguaje, el de la ambigüedad es el principal. Éste aparece cuando alguna unidad lingüística, a saber, sonido, palabra, oración, se puede interpretar en más de una manera [5]. Tal situación se puede notar, a nivel morfológico, por ejemplo, en la palabra "traje" en la oración "traje bebidas" a diferencia de la palabra con la misma forma en la oración "viste de traje*"*.

El PLN incluye a los sistemas traductores automáticos. Mediante ellos, un usuario puede leer, en su propio lenguaje, un texto escrito en otro lenguaje. Asimismo puede escribir en su lenguaje para otros que usan un lenguaje diferente. También es capaz de conversar, de forma escrita u oral, con otros que no comparten su misma lengua [2].

Los lenguajes formales pueden usarse para modelar lenguajes naturales. Para generar lenguajes formales se puede utilizar un procedimiento llamado gramática libre de contexto (GLC), que es equivalente a una forma Backus-Naur (*BNF* por sus siglas en inglés). Una GLC es un conjunto de reglas de "producciones", cada

una de las cuales expresa las maneras en que los símbolos del lenguaje pueden ser agrupados y ordenados unos junto a otros; además de un vocabulario o diccionario de palabras y símbolos. Formalmente, una GLC es una 4-tupla que puede representarse genéricamente como $G = (N, \Sigma, R, S)$. Aquí, N es un conjunto de símbolos no terminales, Σ es un conjunto de símbolos terminales, R es un conjunto de reglas de producción, S es el símbolo inicial de la gramática. El conjunto de reglas de producción está formado por expresiones como $A \rightarrow \beta$. Ésta se puede leer "A deriva en β". Aquí A es la "cabeza de la producción" o "antecedente". Además, en la expresión $A \rightarrow \beta$, el símbolo A es no terminal. β es una cadena de símbolos del conjunto infinito de cadenas $(\Sigma \cup N) *$ [6].

Un lenguaje se define a partir de la "derivación". Una cadena deriva en otra si puede ser reescrita como la segunda mediante la aplicación de una serie de reglas. Formalmente podemos establecer que si $A \rightarrow \beta$ es una producción de P y α y γ son cualquier cadena en el conjunto $(\Sigma \cup N) *$, entonces decimos que $\alpha A \gamma$ "deriva directamente" en $\alpha\beta\gamma$, o bien que $\alpha A \gamma \Rightarrow \alpha\beta\gamma$. Así, la derivación es una generalización de la derivación directa:

Sean $\alpha_1, \alpha_2, \ldots, \alpha_m$ cadenas en el conjunto $(\Sigma \cup N) *$, con $m \geq 1$ tal que $\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m$, entonces decimos que α_1 deriva en α_m, o bien, $\alpha_1 \overset{*}{\Rightarrow} \alpha_m$.

Formalmente se puede definir un lenguaje $\mathcal{L}_G$ generado por una gramática G como el conjunto de cadenas compuestas por símbolos terminales que pueden ser derivados de un cierto símbolo inicial S. Esto es $\mathcal{L}_G = \left\{ w | w \text{ está en } \Sigma * \text{ y } S \overset{*}{\Rightarrow} w \right\}$. Note que en un vocabulario, el símbolo no terminal asociado con cada palabra define una categoría léxica, la categoría de la palabra (sustantivo, artículo, adjetivo, verbo, etc.) [6]. Un componente léxico o *token* es un elemento de una categoría léxica (perro, los, alto, reír, etc.).

Las gramáticas regulares son equivalentes a las expresiones regulares. Las gramáticas regulares pueden ser "*left-linear*" o "*right-linear*". Una regla en una gramática *right-linear* tiene un solo no terminal en la izquierda y a lo más un no terminal en el lado derecho. Si existe un no terminal en el lado derecho, debe ser el último símbolo de la cadena. El lado derecho de una gramática *left-linear* tiene una configuración contraria (el lado derecho debe empezar con máximo un único no terminal). Todos los lenguajes regulares tienen gramáticas *left-linear* y *right-linear* [6].

Es posible realizar diferentes tipos de análisis en PLN, como el morfológico, el sintáctico, el semántico, de discurso, etc. El problema de reconocer que una palabra, por ejemplo, en plural, digamos *zapatos* se puede descomponer en morfemas ("zapato" y "–s") y construir una representación estructurada de tal descomposición, se conoce como análisis morfológico. Un análisis sintáctico o *parsing* es la combinación del reconocimiento de una cadena (oración) de entrada con la asignación a ella de una estructura sintáctica. Se suele representar tal estructura (o derivación) mediante un árbol. Tales árboles comúnmente se muestran invertidos, es decir la raíz en la parte superior. Se necesita especificar los algoritmos mediante los cuales las gramáticas libres de contexto producen los árboles. Un analizador sintáctico o *parser* puede ser visto como como una búsqueda por el espacio de posibles árboles hasta encontrar el árbol correcto que reproduzca una oración dada. La mayor parte de los

analizadores sintácticos utilizan dos estrategias, la búsqueda descendente o búsqueda dirigida por objetivo y la búsqueda ascendente o búsqueda dirigida por datos. La búsqueda descendente recorre un árbol sintáctico de la raíz a las hojas. Por su parte la ascendente lo hace de las hojas a la raíz [6].

Existen herramientas para el desarrollo de analizadores sintácticos como BISON, YACC (Yet Another Compiler-Compiler) y JavaCC (Java Compiler Compiler). Los dos primeros usan salidas en código C y C++ respectivamente, mientras que JavaCC es de código abierto para el lenguaje de programación Java. JavaCC genera analizadores descendentes, permite la construcción de árboles de abajo hacia arriba, hace derivaciones por la izquierda, usa gramática LL(l) y LL(k), una versión de EBNF (Extended Backus-Normal Form), proporciona integración de los analizadores léxicos y sintáctico, permite el uso de *tokens* especiales (a ser procesados por el desarrollador). Se halla entre las que tienen mejor manejo de errores, además de ser una herramienta de nueva generación.

2. Metodología

El sistema es una herramienta de apoyo para el proceso de aprendizaje lingüístico para las personas sordas. Con el fin de identificar los elementos sobresalientes del lenguaje español utilizado en este tipo de enseñanza se realizó una investigación y así se obtuvieron las palabras más usadas. Con ello se definió un determinado léxico. De la misma manera se consultaron, y posteriormente se definieron las estructuras gramaticales más usadas por dicho sector de la población.

El sistema consta de dos módulos, un cliente y un servidor cuya arquitectura se muestra en la Fig. 1.

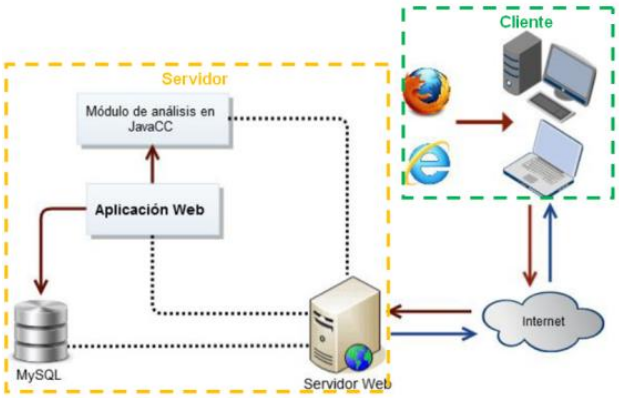

Fig. 1. Arquitectura de alto nivel para el sistema de apoyo lingüístico.

En el servidor se implementa un analizador léxico y uno sintáctico. El primero consiste en validar que las palabras utilizadas se encuentren previamente definidas en una base de datos. Por su parte, el segundo es el que valida que se estén usando, también, sólo las normas de gramática predefinidas. La relación entre los analizadores léxico y sintáctico se esquematiza en la Fig. 2.

El cliente contiene el despliegue de la interfaz gráfica a través del navegador *Web*, en la cual puede tener acceso a dos ejercicios que ayudan a su desarrollo lingüístico. Ambos tipos ejercicios realizan un análisis sintáctico, y uno de ellos también un análisis léxico. El sistema permite al usuario la corrección de la estructura que ha escrito en español, si ésta contiene algún error, hasta, eventualmente obtener una construcción adecuada.

Para comunicar ambos módulos se usa el *framework* Java Server Faces ya que permite separar la vista, o presentación, de la lógica de negocio mediante su patrón de diseño MVC2 (Modelo Vista Controlador). Es definido como un *framework* extensible.

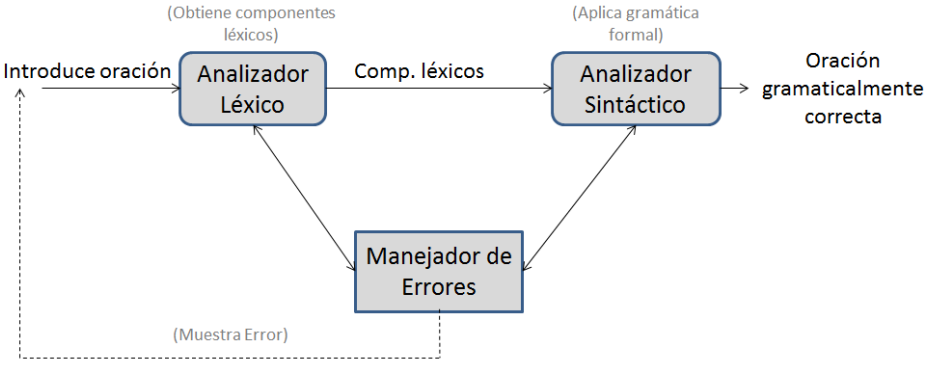

Fig. 2. Método de validación gramatical del sistema.

A continuación describiremos con mayor detalle a los módulos de servidor y de cliente.

Servidor. El servidor se encarga del procesamiento del analizador léxico y sintáctico haciendo uso de JavaCC por sus características lexicográficas y sintácticas requeridas para el análisis de oraciones. Así mismo, se encarga de procesar las llamadas y conexiones al gestor de base de datos MySQL, que es el encargado de almacenar las oraciones y las imágenes que conforman los dos tipos de ejercicios.

Implementación de la gramática formal. Se diseñó una gramática formal a partir del conjunto de estructuras oracionales y vocabulario restringido que, como ya se mencionó, fueron definidas por la investigación. Con el fin de implementar dicha gramática en forma computacional, se utilizaron diferentes herramientas.

JavaCC permite realizar un análisis descendente (genera el árbol sintáctico correspondiente desde la raíz a las hojas) y hacer derivación por izquierda (gramática LL). Para el desarrollo de la gramática LL se hizo uso de la notación EBNF por ejemplo:

```
<forma nominal> := <articulo><sujeto> | <articulo><sujeto><preposicion>
```

Para modelar la gramática se generaron de manera individual los árboles sintácticos de cada una de 15 estructuras y se constituyeron en uno solo. Recuérdese que un sintagma es una unidad lingüística compuesta por una o más palabras a la que también se le ha llamado “frase”. En este sistema se utilizan 5 sintagmas principales **sintagma nominal** cuyo núcleo es el sustantivo o pronombre, **sintagma verbal** con

un verbo como núcleo, **sintagma adverbial** (adverbio como núcleo), **sintagma adjetival** (adjetivo como núcleo) y **sintagma preposicional** (preposición como núcleo). La conjunción sólo realiza la función de enlace, siempre entre sintagmas del mismo tipo. Cada uno de ellos se deriva en componentes más pequeños, e incluso en otros sintagmas, que también cumplen una función específica dentro de una oración obteniendo el árbol que se muestra en la Fig. 3.

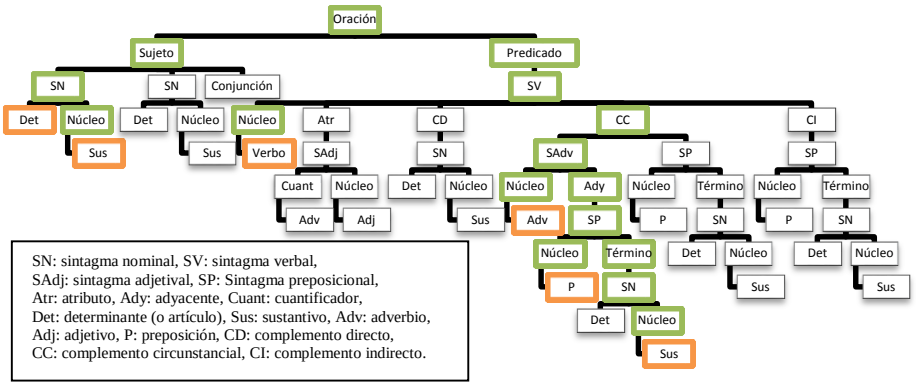

Fig. 3. Representación de la gramática mediante una estructura de árbol.

Analizador léxico. En la especificación léxica, mediante el uso de expresiones regulares, se valida que todas las palabras que conforman la oración correspondan con el vocabulario definido. Se manejan básicamente tres elementos:

- Patrones: las reglas que describen un conjunto de caracteres o palabras.
- Componentes léxicos o *tokens*: símbolos terminales de la gramática.
- Atributos: información adicional para cada componente léxico (relevante para análisis semántico).

En este proceso se leen los caracteres de entrada para verificar un patrón, se obtiene el *token* (carácter a carácter) y se espera la petición del analizador sintáctico para entregar el *token* que ya constituye una palabra.

Configuración léxica. Debido a la delimitación del vocabulario se definió directamente la morfología de cada una de las palabras, considerando sus accidentes gramaticales: género y número. Por ejemplo para artículos tenemos la siguiente definición del léxico mediante sintaxis JavaCC:

```
<ARTICULO:    ("E"|"e")("l")    |    ("L"|"l")("a")    |    |
("L"|"l")("os")  |  ("L"|"l")("as")  |   ("U"|"u")("n")  |
("U"|"u")("na")  |  ("U"|"u")("nos")  |  ("U"|"u")("nas")  >
```

Se analiza el flujo de caracteres clasificándolos en *tokens* (palabras válidas) y asignándoles un componente léxico, verificando el máximo número de caracteres posibles que correspondan a una palabra definida en la sintaxis léxica. Se asignan

valores enteros para cada componente léxico al igual que el atributo `image` mediante el cual se vincula la cadena analizada a un componente definido. En caso contrario se genera y despliega un error.

Analizador sintáctico. Con la especificación sintáctica se corrobora que la cadena de entrada cumpla con una determinada estructura. Es así que es posible identificar si la oración de entrada corresponde a cualquiera de las estructuras definidas. Esto involucra la petición al analizador léxico, examinar el símbolo de la regla, aplicar la derivación y verificar la correspondencia del *token*.

Configuración sintáctica. Se realizó la definición de las reglas gramaticales mediante el formato *EBNF* propio de Java CC, compuesta por elementos terminales (como el espacio, *SPACE*, en el siguiente ejemplo) y elementos no terminales (como el predicado en el siguiente ejemplo) que son derivados hasta encontrar el símbolo final. Veamos el ejemplo:

```
void oración():{}

{

sinNominal() <SPACE> ( predicado() | suj_compuesto()
predicado() | sinPreposicional() <SPACE> predicado() )

}
```

Las estructuras validadas por el analizador son aquellas definidas por la gramática previamente expuesta. Cuando no se encuentra una correspondencia válida se arroja una excepción que indica el error.

Fig. 4. Ejemplo de ejercicio “Formar Oración”.

Cliente. El sistema proporciona al usuario con discapacidad auditiva la selección de dos ejercicios que apoyan su aprendizaje del lenguaje español, los cuales están disponibles en cualquier tipo de navegador Web con conexión al servidor.

Existen dos tipos de actores: el usuario que interactúa con los ejercicios del sistema y el administrador que se encarga de la edición de los ejercicios. Para diferenciar el tipo de actor se ha creado un Inicio de sesión que permite dar los permisos respectivos a cada uno de los usuarios.

Ejercicio "Formar Oración". El primero de los tipos de ejercicios consiste en arrastrar varias cajas de texto hacia un contenedor. Al usuario se le presenta una oración en desorden que cumple con alguna de las 15 estructuras gramaticales definidas (Ver Fig. 4). Dicha oración se encuentra almacenada en la base de datos del servidor, al que se accede de manera remota. Del mismo modo, a través del uso de *servlets*, se hace la comunicación para la verificación de la oración en el analizador léxico-sintáctico del servidor.

Ejercicio "Escribir Oración". El segundo tipo de ejercicio consiste en redactar una oración relacionada con la imagen mostrada. Al usuario se le despliega de manera aleatoria una de las imágenes almacenadas en el servidor, posteriormente debe ingresar una oración relacionada con la imagen, por lo que se permite un uso restringido de caracteres. Una vez ingresada la oración se envía al analizador en el servidor para su análisis, en caso de ser correcta se muestra el acierto como en la Fig. 5 o el respectivo error para una oportunidad más.

Fig. 5. Ejemplo de ejercicio "Escribir Oración".

Módulo de edición. El usuario administrador puede crear, eliminar o modificar algún ejercicio. En el tipo de ejercicio "Formar Oración", el usuario puede ingresar nuevas oraciones o seleccionar alguna de las existentes para su edición o borrado. Del mismo modo el administrador podrá realizar dichas acciones para las imágenes requeridas en el segundo tipo de ejercicio. En ambos casos las modificaciones se realizan desde el navegador del cliente y quedan respaldadas en el servidor.

3. Resultados

En esta sección se muestran los resultados obtenidos en la evaluación durante la ejecución del sistema. Los ejercicios a validar son: Formar oración y Escribir oración.

En el ejercicio "Formar oración" se muestran las palabras desordenadas de manera aleatoria. Supongamos que las palabras presentadas son:

debajo	de	Las	cama	están	pelotas	la

una oración que el sistema valida como correcta es:

Las	pelotas	están	debajo	de	la	cama

ya que corresponde a la estructura gramatical:

Art - Sus - Verb - Adv - Prep - Art – Sus.

Esta estructura tiene el subárbol sintáctico que se muestra de forma resaltada en la Fig. 3.

El sistema no realiza la validación semántica por lo que no estudia el significado de la expresión lingüística. Tomando como válida la siguiente estructura que corresponde a la misma estructura gramatical:

Las	cama	están	debajo	de	la	pelotas

Para el mismo conjunto de palabras el usuario podría introducir la oración:

Las	pelotas	están	debajo	cama	la	de

En este caso la palabra “cama” se encuentra ubicada erróneamente dentro de la estructura de la oración, por lo que se marca en rojo con un texto descriptivo en la aplicación. El texto en el sistema señala para este caso, que “cama" es un sustantivo y que las opciones esperadas son una preposición o contracción según las estructuras gramaticales definidas.

Es necesario que el usuario identifique qué elemento de la oración corresponde a cada componente, es decir, de la oración que se muestra, cuál palabra representa un sustantivo, cuál un verbo, cuál un adjetivo, etc.

En la Fig. 4 se observa que las palabras presentadas son:

profesores	felices	bailan	Los

El usuario arrastra cada una de las palabras y puede formar la oración:

Los	profesores	bailan	felices

la cual corresponde a una estructura gramatical: `Art-Sus-Verb-Adv`, la cual es válida. Para este mismo conjunto de palabras el usuario puede ingresar la oración:

bailan	felices	Los	profesores

la cual presenta una estructura : `Verb-Adv-Art-Sus` que según el árbol sintáctico expuesto en la Fig. 4 es una estructura válida, a pesar de que la palabra "Los" está en mayúscula ocupando el tercer puesto.

Los usuarios de la aplicación son sordos que ya han sido alfabetizados, por lo que deberían conocer los componentes léxicos y su función en la oración, lo que practicarán con la herramienta será manejarlos en conjunto.

Para el segundo tipo de ejercicio, previo al análisis sintáctico antes descrito, se lleva a cabo el análisis léxico. El usuario ingresa una oración relacionada con la imagen. Suponga que para la imagen de la Fig. 5 el usuario ingresa la oración: **"El s3ñor está enojado"** el sistema marcaría que el formato no es válido, ya que únicamente se permite el uso de caracteres. Si se introduce algún dígito o signo ortográfico no permitido (a excepción de acentos) se notifica al usuario, y por lo tanto no se procesa la solicitud de enviar la oración al analizador para su análisis.

Para la oración **"El señior está enojdo"** se puede observar que las palabras **"señior" y "enojdo"** están mal escritas, en este caso el sistema marcaría en naranja "señior" diciendo que dicha palabra no existe o está escrita incorrectamente.

En caso de que el análisis léxico sea correcto se manda a llamar al analizador sintáctico. Por ejemplo para la oración **"El señor está muy enojado"**, el análisis léxico para cada una de las palabras es satisfactorio, posteriormente se procede al análisis sintáctico detectando la estructura gramatical:

```
Art-Sus-Verb-Adv-Adj.
```

Los análisis en la aplicación se realizan de forma secuencial por lo que si el sistema detecta un error lo reporta y detiene el proceso. Si existen dos errores, las sugerencias de corrección solo son con respecto al primer error presentado.

4. Conclusiones

La aplicación detecta los errores a nivel léxico y sintáctico, pero sólo considerando una gramática específica. La validación de oraciones gramaticalmente correctas se realiza satisfactoriamente, sin embargo en algunas ocasiones, tales oraciones, carecen de sentido, por lo que la inclusión del análisis semántico mejoraría el trabajo presentado.

A pesar de que las estructuras gramaticales propuestas eran sencillas, al probarlo con personas con discapacidad auditiva se corroboró la dificultad que para ellos representa expresarse en español escrito; por lo que se podría mejorar contando con diversos niveles de dificultad, así como la inclusión de videos que expliquen con lengua de señas mexicana el ejercicio a realizar.

Referencias

1. Censo de Población y vivienda (Cuestionario ampliado). Instituto Nacional de Estadística, Geografía e Informática (INEGI), http://www3.inegi.org.mx/sistemas/TabuladosBasicos/Default.aspx?c=27303&s=est (2010)
2. Gelbukh, A.: Procesamiento de lenguaje natural y sus aplicaciones. Komputer Sapiens, Vol. 1, pp. 6–11 (2010)
3. Quesada Moreno, J.F., de Amores Carredano, J.G.: Diseño e implementación de sistemas de traducción automática. Universidad de Sevilla, Sevilla (2000)

4. Alfonso Galipienso, M.I., Cazorla Quevedo, M.A., Colomina Pardo, O., Escolano Ruiz, F., Lozano Ortega, M.Á.: Inteligencia artificial: modelos, técnicas y áreas de aplicación. Paraninfo, Madrid Tesis (2003)
5. Gelbukh, A., Sidorov, G.: Procesamiento automático del español con enfoque en recursos léxicos grandes. Dirección de Publicaciones del IPN, México (2010)
6. Jurafsky, D., Martin, J.H.: Speech and Languague Processing, an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, New Jersey (2008)