GENERADOR DE CÓDIGO AUTOMÁTICO (GCA) EN LENGUAJE JAVA A PARTIR DE CON CONJUNTO DE INSTRUCCIONES EN LENGUAJE NATURAL

Hugo Francisco Adan Oliva[1], Miguel Angel Peña López[2], Ruth Saez de Nanclares Rodríguez[3], Martha Alicia Rocha Sánchez[4].

RESUMEN

En este trabajo presentamos un sistema de traducción experimental, entre lenguaje natural español y lenguaje de programación Java, basado en modelos y técnicas de traducción automática estadística. Para la generación del modelo se utilizó un conjunto restringido de frases en español y Java. El modelo encontrado presenta la capacidad de los traductores basados en estadística, donde se parean frases en un idioma (español) con su correspondiente en otro (Java).

Para el desarrollo, se utilizaron herramientas open source (*Moses, GIZA++ e IRSTLM)* que generaron los modelos estadísticos de los lenguajes y posteriormente dieron forma al traductor. Las métricas *BLEU* (Papineni et al. 2002) y *WER* (Thoma, 2003) nos ayudaron a evaluar los modelos generados, y así, determinar con certidumbre un modelo óptimo.

PALABRAS CLAVE Traducción automática estadística, procesamiento de lenguaje natural, lenguaje formal, decoder, sistema de traducción automática estadística.

[1] Ingeniería en Sistemas Computacionales, Instituto Tecnológico de León. Avenida Tecnológico S/N - Fraccionamiento Industrial Julián de Obregón, C.P. 3729, Guanajuato, León, (477) 710 5200.

[2] Ingeniería en Sistemas Computacionales, Instituto Tecnológico de León. Avenida Tecnológico S/N - Fraccionamiento Industrial Julián de Obregón, C.P. 37290, Guanajuato, León, (477) 710 5200.

[3] Profesora, Instituto Tecnológico de León, Departamento de Sistemas y Computación, Avenida Tecnológico S/N - Fraccionamiento Industrial Julián de Obregón, C.P. 37290, Guanajuato, León, Teléfono (477) 710 5200; Fax (477) 711 2072; r.saezdenanclares@yahoo.com.mx

[4] Profesora, Instituto Tecnológico de León, División de Estudios de Posgrado e Investigación, Avenida Tecnológico S/N - Fraccionamiento Industrial Julián de Obregón, C.P. 37290, Guanajuato, León, Teléfono (477) 710 5200; Fax (477) 711 2072; marthaalicia.rocha@itleon.edu.mx

INTRODUCCIÓN

El Procesamiento del Lenguaje Natural (PLN) es un área de investigación y aplicación de la inteligencia artificial, que explora cómo las computadoras pueden ser usadas para entender y manipular textos en lenguaje natural o el habla para hacer cosas útiles. La investigación del PLN tiene como objetivo, reunir conocimientos sobre cómo los seres humanos entienden y usar el lenguaje, para modelar computacionalmente estos patrones y reducir el umbral entre la comunicación humano – computadora (Chowdhury, G., 2003). Una subárea del PLN es la traducción automática y de acuerdo a (Armentano-Oller, Carme, et al. 2007), *"La traducción automática es la obtención de un texto equivalente (esto es, que preserve el contenido) en una lengua destino a partir de un texto en una lengua origen. Estamos hablando, en cualquier caso, de lenguas naturales (como, por ejemplo, el español, el inglés), no de lenguajes informáticos (como, por ejemplo, Java o XML); la traducción entre lenguajes informáticos es una cuestión totalmente diferente (aunque puede compartir algunas técnicas básicas con la traducción automática) que se aborda en el campo del diseño de compiladores."*

Un lenguaje de programación se define como un método de escritura particular y restringida, que describe estructuras concretas. Los primeros lenguajes de programación fueron rústicos y complejos, constaban de escribir cadenas de ceros y unos (binario) para construir programas, lo que lo volvía una tarea bastante compleja, tal que, se necesitaba ser un experto para poder hacerlo. Bajo este principio, comenzó el desarrollo nuevos lenguajes, utilizando juegos de caracteres, buscando asemejarse al lenguaje humano, con ello se comenzó a facilitar la escritura de programas. Hoy en día algunos de los lenguajes de programación son semejantes al lenguaje humano, sin embargo, no tienen la misma flexibilidad, ya que no permiten ambigüedad debido a que su escritura se basa en gramáticas formales libres de contexto (Jurado, 2008). La traducción entre lenguajes de programación, utiliza técnicas que convierten un programa escrito en un lenguaje origen, a un texto equivalente de un lenguaje objeto, produciendo en algunos casos, mensajes de error durante la traducción. Además, los traductores de lenguajes formales engloban a los compiladores, donde el lenguaje destino es código máquina, y los intérpretes en los que el lenguaje destino se constituyen por las acciones atómicas que puede ejecutar el intérprete (Gálvez Rojas et al., 2005).

La traducción automática estadística, es una técnica de traducción entre idiomas y usa datos lingüísticos tradicionales. Se fundamenta en el principio de hacer alineaciones entre frases, grupos de palabras y palabra a palabra, de textos paralelos escritos en dos idiomas (fuente y destino), para calcular probabilidades de que una palabra o frase de una lengua, se corresponda con una palabra o frase de la lengua paralela con la que está alineada. La característica principal de estos sistemas de traducción es, contar con un corpus (pares de frases entre lenguajes) lo bastante robusto, que permita una mayor base de conocimiento y aumente la generación de probabilidades (Tertoolen R., 2012) (Oliver et al., 2005).

Las primeras ideas de traducción automática estadística fueron introducidas por Warren Weaver en 1949 (W. Weaver, 1955). La traducción automática estadística fue reintroducida en 1991 por investigadores de la Thomas J. Watson Research Center de IBM y ha contribuido al resurgimiento significativo del interés por la traducción automática en los últimos años.

Las ideas que hay detrás de la traducción automática estadística vienen esencialmente, de la probabilidad $\boldsymbol{p(e|f)}$ de que una cadena $\boldsymbol{e}$ de la lengua nativa (por ejemplo, inglés) sea la traducción de una cadena $\boldsymbol{f}$ en la lengua extranjera (por ejemplo, francés). Generalmente, estas probabilidades se calculan utilizando técnicas de estimación de parámetros.

El Teorema de Bayes se aplica a $\boldsymbol{p(e|f)}$, la probabilidad de que la cadena del idioma extranjero produzca la cadena nativa para conseguir $\boldsymbol{p(e|f)\ \alpha\ p(f|e)\ p(e)}$, donde el modelo de traducción $\boldsymbol{p(f|e)}$ es la probabilidad de que la cadena nativa sea la traducción de la cadena extranjera, y el modelo de lengua $\boldsymbol{p(e)}$ es la probabilidad de ver aquella cadena nativa. Matemáticamente hablando, encontrar la mejor traducción $\boldsymbol{e}$ se consigue escogiendo aquella que dé la probabilidad más alta, y se obtiene al calcular la ecuación 1).

$$\tilde{e} = \arg\max_{e \in e^*} p(e|f) = \arg\max_{e \in e^*} p(f|e)p(e) \quad \textbf{(1)}$$

TRABAJOS RELACIONADOS

Dentro de los trabajos y propuestas sobre la generación de código automático que parten del análisis de lenguaje natural encontramos: *UN-Lencep* (Zapata Jaramillo et al. 2010), herramienta de software que genera automáticamente código Java y PHP. Este trabajo se clasifica dentro de las herramientas CASE utilizadas por analistas de requerimientos para un sistema. El trabajo se centra en la generación de plantillas de clases (PHP y JAVA) para cada concepto del lenguaje natural ingresado, junto con los atributos que éste contiene, y las relaciones con los demás conceptos.

El proyecto LNE2SQL (Reyes García, González García, 2013) consta del desarrollo de un sistema de traducción automática de sentencias en español a instrucciones de SQL para hacer consultas a una base de datos. En esta investigación el traductor procesa las consultas escritas en español sobre cualquier dominio a SQL e implementa un mecanismo de expansión de consultas a partir de un diccionario de dominio creado con la ayuda de la base de datos léxica WordNet.

(Muellner, James, 1995) muestra un traductor de frases en inglés a código COBOL. Este sistema utiliza un diccionario de palabras y reglas para traducir, por palabra, frase y estructura de la oración en una especificación del programa con declaraciones válidas de COBOL. Se imponen restricciones de dominio para el sistema, tal que, si se encuentra una palabra desconocida, el sistema solicita interactivamente la información al usuario para crear una entrada de diccionario. El sistema permite la flexibilidad de entrada, así como la verificación de las traducciones, además permite modificar los diccionarios del sistema para que coincida con su estilo de escritura.

Nuestro proyecto comprende un dominio de traducción distinto a los encontrados, buscando innovar en el proceso de traducción de un lenguaje natural a un lenguaje formal, además de minimizar las brechas que existen en el desarrollo de software de base, y pudiendo dar un futuro enfoque al proyecto, aplicándolo a la enseñanza/aprendizaje del lenguaje Java.

MÉTODOS Y MATERIALES

Para llevar a cabo el desarrollo de nuestro proyecto el método que utilizamos fue deductivo-experimental en donde partimos de la hipótesis siguiente:

Por medio de un modelo óptimo de traducción automática estadística, donde una frase en lenguaje natural español (variable independiente *x*) es capaz de generar una estructura en lenguaje de programación Java (variable dependiente *y*).

Adquisición de las muestras de aprendizaje o corpus. Consistió en etiquetar pares de frases entre lenguaje natural español y sentencias de lenguaje Java (corpus). Se escribió un corpus inicial de 5190 frases pareadas entre el español y Java. Este corpus inicial tuvo correcciones y fue revisado por 3 expertos en programación Java quienes corrigieron manualmente las traducciones. Debido a la gran cantidad de notaciones y simbologías que utilizan los lenguajes en su escritura, se vuelve necesario hacer un preproceso a los archivos que contienen los pares de frases en español y Java.

Se crearon posteriormente, los modelos de traducción con *Moses* (Koehn et al. 2007)[5], para el preproceso de las frases del corpus se *tokenizó*, en este paso se identifica y separa a cada componente léxico (token). La tokenización se aplica a ambos archivos que conforman el corpus y como resultado se generan dos nuevos archivos que contienen el texto original tokenizado. Otro preproceso para la diferenciación de mayúsculas y minúsculas es el *truecasing*. Esta fase consta de dos pasos: 1) Se extrae las estadísticas de aparición de tokens de los archivos tokenizados, que se almacenan en dos nueos archivos, ahí se encuentra el número de veces que aparece cada token diferenciando mayúsculas de minúsculas; 2) Crear dos archivos (español y Java) unificando y reduciendo el ruido en ellos, los deja en un formato estándar, sustituye los tokens por su caso en minúscula si es que existe. El último paso de preparación del corpus es la limpieza. Este proceso consta de limitar el tamaño de las frases (cantidad de palabras) y eliminar las frases vacías.

La generación de los modelos de lenguaje se hizo con la herramienta IRST Language Modelling Toolkit (IRSTLM) (Federico et al., 2007), que genera el modelo en base a ngramas. Este proceso se lleva a cabo en dos pasos. El primero identifica los ngramas que existen dentro del corpus para el lenguaje objetivo (Java) y el segundo paso genera y asigna números estadísticos a cada token que utilizará el decoder para asociar los lenguajes. El entrenamiento de los modelos se logró con la herramienta GIZA++ (Och, Ney, 2003), se encarga de hacer la alineación a nivel frase y palabra de los textos paralelos de lenguajes, el proceso de entrenamiento de extracción de frases se calcula con el *Moses* a través de la simetrización de matrices (Koehn et al. 2007). La tabla de frases $(\boldsymbol{p}(\boldsymbol{e}|\boldsymbol{f}))$ es el modelo que se utiliza por el decoder en combinación con el modelo del lenguaje $(\boldsymbol{p}(\boldsymbol{e}))$ es decir, es el sistema de traducción entrenado y listo para ser usado.

La eficiencia de los modelos se midió utilizando dos métricas:

[5] http://www.statmt.org/moses/?n=Development.GetStarted

A) El W*ord Error Rate* (WER), basado en la distancia-Levenshtein, consta de encontrar el número mínimo de sustituciones, eliminaciones e inserciones que tienen que llevarse a cabo para convertir una frase de prueba (hipótesis), que se compara con el texto objetivo (referencia). Un coeficiente WER pequeño indica un alto nivel de efectividad y certeza del traductor, a mayor WER, menor efectividad (Thoma, 2003).

B) La medida automática BLEU. Este criterio calcula la media geométrica de la precisión de n-gramas de diversa longitud entre una hipótesis y un conjunto de traducción de referencia multiplicado por un factor de BP (•) que penaliza frases cortas, el coeficiente BLEU se calcula mediante la ecuación 2).

El coeficiente denota la precisión de los ngramas en la traducción de la frase tomada como hipótesis (Papineni et al. 2002).

$$Bleu = BP(*) * \exp \sum_{n=1}^{n} \frac{\log Pn}{N} \quad \textbf{(2)}$$

RESULTADOS

Al momento se han creado 11 corpus distintos con su respectivo sistema de traducción, el proceso de generación de modelos del lenguaje se hace comúnmente mediante la interfaz de comandos, pero optamos por escribir scripts en Python para agilizar estos procesos.

Conforme escribimos los corpus y probamos los sistemas, pudimos notar que las herramientas diseñadas para traducción entre idiomas, permiten un grado de libertad limitado al momento de hacer traducción entre un lenguaje natural y un lenguaje formal. La tabla 1 muestra los resultados de la evaluación con WER y BLEU para cada corpus, se identifica con un número a cada corpus escrito, además muestra las cantidades de palabras en español y Java utilizadas, así como el total de frases y el valor de WER y BLEU.

Tabla 1. Evaluación cruzada con métricas WER y BLEU

No. De corpus	Total palabras español	Total palabras Java	WER
1	158	18	1.546154
2	75	77	4.084615
3	69	60	4.084615
4	179	18	4.042308
5	154	17	4.138462
6	151	22	2.776432
7	386	22	1.676071

8	310	22	1.328461
9	368	22	4.007692
10	319	22	3.003479
11	344	27	1.363879
12	350	29	4.769212

CONCLUSIONES

Los corpus desarrollados han mostrado resultados aceptables en sus evaluaciones WER, esto nos da la pauta para seguir con los experimentos con el sistema de modelos de lenguaje.

No se ha llegado al modelo de lenguaje óptimo, seguirá el proceso de mejora continua, basándose en las métricas arrojadas.

REFERENCIAS

Muellner, Robert James (1995); Interactive automatic generation of COBOL code from English specifications; Thesis Ph.D.; Illinois Institute of Technology; United State – Illinois; Document URL http://search.proquest.com/docview/304194899;
S. Vogel, H. y Ney C. Tillmann. 1996. Basados en HMM Palabra Alineación en StatisticalTranslation. En Coling '96: La 16 ª Conferencia Internacional de Lingüística Computacional, pp. 836-841, Copenhague, Dinamarca.
Papineni, K., Roukos, S., Ward, T., and Zhu, W. J. (2002). "BLEU: a method for automatic evaluation of machine translation" in ACL-2002: 40th Annual meeting of the Association for Computational Linguistics pp. 311—318.
Chowdhury, G. (2003) Natural language processing. Annual Review of Information Science and Technology, 37. pp. 51-89. ISSN 0066-4200
Och Franz Josef, Hermann Ney. "A Systematic Comparison of Various Statistical Alignment Models", Computational Linguistics, volume 29, number 1, pp. 19-51 March 2003.
Gálvez Rojas Sergio, Miguel Ángel M. Mata (2005). Java a tope: Traductores y compiladores con Lex/Yacc, JFlex/Cup y JavaCC. Dpto. de Lenguajes y Ciencias de la Computación E.T.S. de Ingeniería Informática Universidad de Málaga.
Oliver Antonio, Toni Badia, Gemma Boleda, Maite Melero. (2005). Traducción automática estadística basada en *n*-gramas. Núm. 35, pp. 77-84. Universitat Pompeu Fabra. Barcelona, España.
Armentano-Oller, Carme, et al. "Apertium, una plataforma de código abierto para el desarrollo de sistemas de traducción automática". En: Proceedings of the FLOSS International Conference 2007 [Recurso electrónico] / J. Rafael Rodríguez Galván, Manuel Palomo Suarte (coords.). Cádiz: Servicio de Publicaciones de la Universidad de Cádiz, 2007. ISBN 978-84-9828-124-8, pp. 1-16. http://rua.ua.es/dspace/bitstream/10045/27531/1/armentano07.pdf
Federico, Marcello and Cettolo, Mauro. Proceedings of the Second Workshop on Statistical Machine Translation (2007). Efficient Handling of N-gram Language Models for Statistical Machine Translation. Association for Computational Linguistics, pp 88-95. http://www.aclweb.org/anthology/W/W07/W07-0712.
Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., & Herbst, E. (2007, June). Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions (pp. 177-180). Association for Computational Linguistics.
Jurado Málaga Elena (2008). Teorías de Autómatas y Lenguajes Formales. Extremadura, España. Edita Universidad de Extremadura. Servicio de Publicaciones.
Zapata Jaramillo C. M., Chaverra Mojica J. J., Zapata Ceballos B. (2010). Generación Automática de Código a Partir del Lenguaje Controlado UN-Lencep. Sistemas, Cibernética e Informática, Medellin Antioquía, Colombia.
Thoma Martin. (2013). Word Error Rate Calculation. Recuperado de http://martin-thoma.com/word-error-rate-calculation/

Reyes García F., González García J. A. (2013). LNE2SQL: Traductor de consultas del lenguaje natural a SQL v2.0. Universidad de las Ciencias Informáticas, La Haba, Cuba.