CONSTRUCCIÓN DE UN CORRECTOR ORTOGRÁFICO HÍBRIDO PARA EL CHABACANO DE ZAMBOANGA[1]

BUILDING A HYBRID SPELL CHECKER FOR ZAMBOANGA CHAVACANO

Marcelo-Yuji Himoro
Universidad Nacional de Educación a Distancia (UNED)
Antonio Pareja-Lora
Universidad de Alcalá / ATLAS (UNED)

RESUMEN
El chabacano zamboangueño es una variedad de criollo español hablada en la ciudad de Zamboanga, Filipinas. Dadas (a) la naturaleza mixta de la lengua, que engloba principalmente elementos del castellano, de las lenguas visayas, del tagalo y del inglés, y (b) su peculiar ortografía de orientación etimológica, los hablantes se ven obligados a manejar diferentes sistemas de escritura. Las variadas grafías prenormativas y los fenómenos de cambio y mezcla de código son también una barrera para la detección y la corrección automática de los errores ortográficos. Este trabajo propone el desarrollo de un corrector ortográfico escalable, interoperable y capaz de dar respuesta al *problema* ortográfico del zamboangueño. Los resultados demuestran que es viable el uso de Hunspell, la tecnología de corrección ortográfica actual por defecto, por medio de su combinación con técnicas de aprendizaje automático y la incorporación de datos del tagalo y del inglés.

PALABRAS CLAVE: chabacano, Zamboanga, corrector ortográfico, error ortográfico, Filipinas

ABSTRACT
Zamboanga Chavacano is a variety of Philippine Creole Spanish spoken mainly in Zamboanga City. Based mostly on Spanish with elements of Visayan, Tagalog and English origins, its mixed nature and peculiar etymology-based orthography forces speakers to deal with different writing systems in order to be able to correctly write the language. The diversity found in the (still in use) non-standard writing systems and the omnipresence of code-switching and code-mixing phenomena are a further challenge for automated spelling error detection and correction. This research aims to develop a scalable and interoperable spell checker, capable of handling the Zamboangueño orthographic problem. Our results show that it is possible to raise the de facto standard spell checker Hunspell performance to acceptable precision levels, by combining it with machine learning techniques and incorporating Tagalog

[1] Agradecimientos: Esta investigación ha sido realizada en parte como fruto de la red de investigación asociada a "LITHME" (*COST Action* CA19102), financiada con fondos del Programa Marco Horizonte 2020 de la Unión Europea. Nuestra más profunda gratitud a R. David Zorc por facilitarnos el acceso al lexicón utilizado en Miravite et al. (2009). Agradecemos también a los grupos de investigación FITISPos (UAH) y ATLAS (UNED) su apoyo continuo en el desarrollo de esta investigación.

CC BY NC ND

and English data in its processing. Such a spell checker would also allow users to indirectly familiarize with the orthography.

KEYWORDS: Chavacano, Zamboanga, spell checker, spell error, Philippines

1. INTRODUCCIÓN

'Chabacano' es un término paraguas que agrupa las diferentes variantes de criollo español habladas en el archipiélago filipino. De todas ellas, el zamboangueño es la que goza de mayor vitalidad y la única que todavía observa un crecimiento natural en su número de hablantes. Usado sobre todo en la ciudad de Zamboanga, en 2010 contaba con 405.798 hablantes nativos en todo el país (*National Statistics Office* 2014), sin contar los numerosos hablantes no nativos y una considerable diáspora en el extranjero. Es, además, la única variante incluida en el programa MTB-MLE (*Mother Tongue-Based Multilingual Education*, o Educación Multilingüe Basada en la Lengua Materna), introducido con la reforma educativa del 2012, y desde entonces es lengua vehicular y asignatura obligatoria de los tres primeros cursos de primaria en las escuelas públicas de la ciudad de Zamboanga (*Government of the Philippines* 2011).

Su reintroducción en la educación animó al gobierno local de la ciudad a iniciar un proceso de codificación y normativización del zamboangueño[2], lo que se plasmó en la aprobación de una ortografía (*Zamboanga Chavacano Orthography*) en 2014 y dió lugar a numerosas iniciativas y publicaciones escritas para su promoción. Asimismo, se usa activamente, aunque de manera limitada, en los medios de comunicación (televisión, radio, prensa, etc.). Pese a esta coyuntura, relativamente favorable, hay evidencias de multiglosia y de pérdida tanto de ámbitos como de frecuencia de uso de la lengua, además de inseguridad entre los hablantes más jóvenes, bien al hablar, bien al escribir (Himoro, 2019). El proceso de normalización en curso es, por lo tanto, vital para permitir que la lengua se utilice de manera efectiva en registros escritos más altos y contrarrestar la hegemonía de las lenguas oficiales (v.g., el inglés y el tagalo).

La aprobación de la ortografía, si bien es un paso muy importante, ha planteado problemas respecto a su aplicabilidad (Himoro y Pareja-Lora, 2020, en prensa). El principio fundamental de la ortografía es que las palabras deben escribirse de acuerdo con su forma en la lengua de origen (DepEd Zamboanga City Division, 2016), lo que implica 1) la necesidad de manejar diferentes sistemas de escritura a la vez y 2) que la ortografía considerada correcta no siempre refleja la evolución fonética de la lengua y, por lo tanto, requiere un gran esfuerzo de memorización por parte de sus usuarios. El objetivo de este trabajo es, precisamente, ayudar a los hablantes de cbk-PH-09 a familiarizarse con la ortografía de la lengua y a ganar una mayor seguridad en su escritura mediante el desarrollo de un corrector ortográfico automático de

[2] De aquí en adelante, nos referiremos al chabacano zamboangueño mediante el identificador 'cbk-PH-09'. Este identificador cumple con las directrices de la recomendación RFC 4646 (https://tools.ietf.org/html/rfc4646), que proporciona un conjunto de buenas prácticas para la construcción de etiquetas identificativas de lenguas. El chabacano aparece en la lista de códigos de la norma ISO 639-3 (https://iso639-3.sil.org/code/cbk) como 'cbk'. Para identificar específicamente la variante zamboangueña, es posible añadir el sufijo 'PH-09' de la norma ISO 3166 (https://www.iso.org/obp/ui/#iso:code:3166:PH). Este sufijo identifica la 'Península de Zamboanga (Región IX)' y abarca tanto la ciudad de Zamboanga como la ciudad de Isabella.

CC BY NC ND

precisión adecuada, interoperable con los principales programas y aplicaciones informáticas más usados por sus hablantes.

El resto del artículo está organizado de la siguiente manera: en el apartado 2 se presentan los trabajos anteriores o relacionados con éste; el apartado 3 describe la hipótesis, los objetivos y la metodología de investigación; el apartado 4 proporciona una descripción de la implementación de las aproximaciones de corrección ortográfica desarrolladas, además de los resultados obtenidos; en el apartado 5 se analiza cada una de las aproximaciones tanto de manera independiente como integrada y se discuten sus ventajas e inconvenientes; finalmente, el apartado 6 aborda las conclusiones finales del trabajo y da cuenta de posibles trabajos futuros.

2. TRABAJOS PREVIOS O RELACIONADOS

En este apartado, presentamos algunos trabajos previos o relacionados con la presente investigación: las principales aproximaciones a la corrección ortográfica (semi-)automática (subapartado 2.1) y los estudios sobre los errores ortográficos cometidos por usuarios del cbk-PH-09 (subapartado 2.2) existentes.

2.1. Principales aproximaciones a la corrección ortográfica (semi-)automática de textos

En primer lugar, la mayor parte de los correctores ortográficos tradicionales está íntimamente relacionada con la creación de las métricas de distancia de edición de Damerau-Levenshtein (Damerau, 1964; Levenshtein, 1966). Esta métrica calcula el número de operaciones básicas (bien la inserción, eliminación o sustitución de un carácter o bien la transposición de dos caracteres adyacentes) necesarias para transformar una cadena de caracteres en otra. No obstante, si bien esta aproximación es la más antigua y adoptada a día de hoy, no es la única. Algunas posibles alternativas son los algoritmos fonéticos (subapartado 2.1.2) y las aproximaciones basadas en la inteligencia artificial o el aprendizaje automático (subapartados 2.1.3 y 2.1.4).

2.1.1. Correctores ortográficos basados en la distancia de edición

Uno de los primeros correctores ortográficos de la historia – si no el primero – fue SPELL (Gorin 1972), basado en la heurística de las métricas de distancia de edición, específico para el inglés y programado en lenguaje ensamblador. En el año 1983, SPELL se portó al lenguaje C bajo el nombre de Ispell (Willisson, 2021), adoptando una estructura más modular, que permitió independizarlo de la lengua.

A partir de Ispell, surge así toda una familia de motores de corrección ortográfica de código abierto, independientes de la lengua, basados en diccionarios y que permiten la codificación opcional de reglas de derivación mediante un fichero externo: GNU Aspell en 1998 (Atkinson, 2019), MySpell en 2002 y Hunspell en 2004 (Németh, 2004), con una limitada retrocompatibilidad entre sí.

Hunspell es, hoy por hoy, el corrector ortográfico por defecto en conocidas tecnologías de código abierto, como los paquetes ofimáticos LibreOffice y OpenOffice, ciertos navegadores (Mozilla Firefox, Google Chrome y otros) y está presente también en paquetes de software propietario, como memoQ o SDL Trados (Németh, 2016). Las dos operaciones más importantes que realiza Hunspell son la detección de errores ortográficos y la generación de candidatos de corrección.

BY NC ND

2.1.2. Algoritmos fonéticos

Un algoritmo fonético consta de una serie de reglas que codifican bajo una misma representación palabras homófonas o cuasihomófonas, de manera aproximada en este último caso. Los algoritmos de este tipo más conocidos son Soundex (Russell, 1918) y Metaphone (Philips, 2000). Pese a su uso inicial en catalogación y sistemas de gestión de base de datos, estos algoritmos también se han aplicado extensivamente en trabajos de investigación sobre la corrección ortográfica y normalización (Gao et al., 2010; Han et al., 2013). De hecho, aparecen integrados en ciertos correctores ortográficos de código abierto (i.e., Aspell y Hunspell, véase el apartado 2.1.1).

Aunque fueron originalmente diseñados para el inglés, se han adaptado también a otras lenguas ya sea mayoritarias, como el portugués, el castellano o el ruso, ya sea más minoritarias, como es el caso del malabar, del alsaciano, del bengalí o del sindi. Estas adaptaciones se han obtenido mediante la formulación manual de reglas similares al caso inglés, y han sido aplicadas, por ejemplo, en los trabajos de UzZaman y Khan (2005), Avanço et al. (2014), Bhatti et al. (2014) o Bernhard y Steiblé (2015).

2.1.3. Traducción automática estadística propiamente dicha y de caracteres

La traducción automática estadística (TAE) parte del principio básico (Poibeau, 2017) de que es posible encontrar equivalencias de traducción (para palabras o conjuntos de palabras) entre una lengua fuente (LF) y una lengua meta (LM) de manera automática mediante el análisis estadístico de textos bilingües (véase la Figura 1).

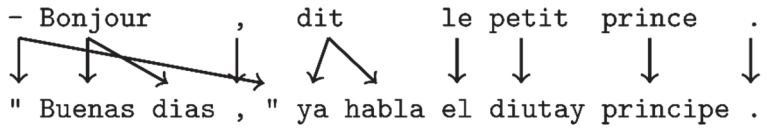

Figura 1. Ejemplo de alineamiento léxico en un sistema tradicional de Traducción Automática Estadística (fuente de las frases: Saint-Exupéry, 1943: 77; Saint-Exupéry, 2018: 66)

Por contra, la traducción automática estadística de caracteres (TAEC) es una variante de la TAE que alinea un carácter o un conjunto de caracteres de la LF con otro carácter o conjunto de caracteres de la LM (Vilar et al., 2007). Así pues, la unidad básica de traducción es el carácter (no la palabra), y los conjuntos de caracteres se convierten en unidades complejas para formar palabras o patrones, delimitados por los caracteres *'#'* y *'$'*, que codifican respectivamente las posiciones inicial y final de la palabra.

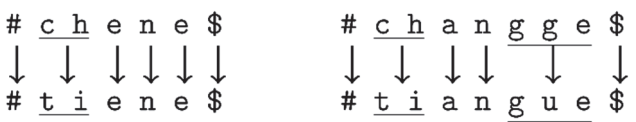

Figura 2. Dos ejemplos de alineamiento en un sistema de Traducción Automática Estadística de Caracteres

BY NC ND

La TAEC surgió originalmente como una alternativa para la TAE entre lenguas relacionadas (Nakov y Tiedemann, 2012; Tiedemann, 2009; Vilar et al., 2007). Posteriormente, sin embargo, ha empezado a aplicarse en tareas ajenas a la traducción, como la transliteración (Karimi, 2008; Tiedemann y Nabende, 2009), la generación de cognados (Beinborn et al., 2013), la corrección ortográfica de términos de búsqueda (Hasan et al., 2015) o la normalización de textos dialectales (Scherrer et al., 2016) e históricos (Korchagina, 2017; Schneider et al., 2017) con notable éxito.

Una manera sencilla de desarrollar un sistema de TAE es utilizar el paquete Moses (Koehn et al., 2007). Este paquete contiene todas las herramientas necesarias para entrenar un modelo, siendo necesario tan solo disponer de (1) datos paralelos en la LF y en la LM; (2) un alineador de textos y (3) un modelo de lenguaje generado a partir de los datos de entrenamiento de la LM.

2.1.4. Redes de neuronas

Las redes de neuronas son una familia de modelos de aprendizaje automático inspirados en el cerebro humano que utilizan métodos de computación no lineales. Son considerados punteros en muchas tareas de Procesamiento del Lenguaje Natural (PLN). Así, hay diferentes trabajos que proponen el uso de redes de neuronas en la corrección ortográfica (Xie et al., 2016; Ghosh y Kristensson, 2017; Etoori et al., 2018) y en la normalización de textos (Ikeda et al., 2016; Tursun y Cakici, 2017).

En primer lugar, Xie et al. (2016) presenta una aproximación a la corrección de textos (de un foro de aprendices de inglés) basada en redes de neuronas, capaz de sortear el problema que suponen las palabras no incluidas en los datos de entrenamiento.

En segundo lugar, Ghosh y Kristensson (2017) proponen asimismo la aplicación de un sistema de redes de neuronas para la corrección y compleción automática de textos.

En tercer lugar, Etoori et al. (2018) muestran cómo construir un corrector ortográfico (basado en una arquitectura de redes de neuronas) para lenguas con menos recursos (*resource-scarce languages*), centrándose en particular en las lenguas índicas, que aplican en concreto a la corrección automática del hindi y el telugu.

Finalmente, Ikeda et al. (2016) y Tursun y Cakici (2017) utilizan un modelo de codificador-decodificador neuronal para normalizar textos en japonés y uigur, respectivamente.

Sin embargo, el problema que presentan las redes de neuronas con respecto a nuestro trabajo es que son más difíciles de integrar con otras soluciones, dada la gran opacidad de sus resultados (se comportan como una caja negra).

2.1.5. Aproximaciones híbridas

En lo que se refiere a aproximaciones híbridas, no se conocen a fecha de hoy trabajos que apliquen de manera conjunta el aprendizaje automático con los correctores ortográficos tradicionales sin afectar la escalabilidad o la interoperabilidad que estos últimos proporcionan.

2.2. Tipologías de errores del cbk-PH-09

En este trabajo, adoptamos como referencia la tipología de errores ortográficos del cbk-PH-09 desarrollada por Himoro y Pareja-Lora (2020). Por falta de espacio, en este apartado se presentan únicamente los aspectos esenciales de la misma para la

CC BY NC ND

comprensión de este trabajo. La Figura 3 muestra la tipología completa.

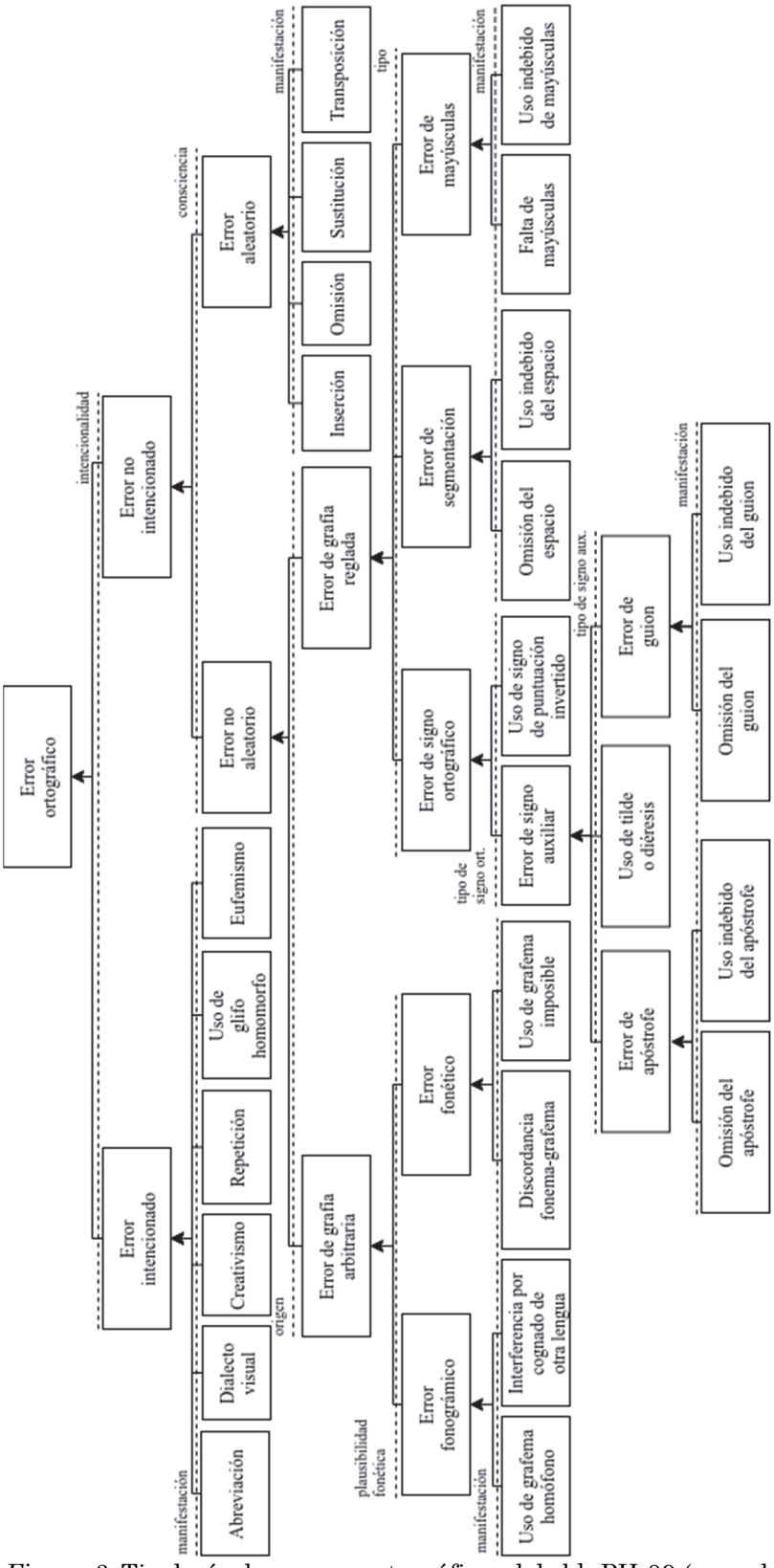

Figura 3. Tipología de errores ortográficos del cbk-PH-09 (reproducción de Himoro y Pareja-Lora, en prensa)

BY NC ND

En un nivel superior, los autores dividen los errores ortográficos en cbk-PH-09 según su intencionalidad. Los 'Errores intencionados' son aquellos que, no siendo errores propiamente dichos en la mayoría de casos, se producen de manera consciente, como pueden ser los 'textismos' (jerga típica de los SMS/mensajes instantáneos intercambiados por internet) o los eufemismos (deformación intencional de una palabra malsonante o blasfema que genera una palabra inexistente). Los 'Errores no intencionados', a su vez, son los que se producen por razones diversas: se subdividen en 'Errores aleatorios', erratas fortuitas o fruto del descuido a la hora de teclear, y 'Errores no aleatorios'. Estos últimos, a su vez, pueden ser 'Errores de grafía arbitraria', que se producen por interferencia de otras lenguas o (sobre todo) por desconocimiento de la etimología, o 'Errores de grafía reglada', que se producen por desconocimiento de las reglas que regulan ciertos aspectos de la lengua.

3. HIPÓTESIS, OBJETIVOS Y METODOLOGÍA DE INVESTIGACIÓN

Por un lado, la hipótesis de investigación principal de este trabajo es que es posible construir un corrector semiautomático con una precisión aceptable y optimizada para el cbk-PH-09 y que constituya una solución escalable para este problema, además de interoperable con los principales programas y aplicaciones informáticas más usados por sus hablantes (navegadores, editores de texto, redes sociales, aplicaciones móviles para el intercambio de mensajes, etc.). Este corrector debe, por lo tanto, ser capaz de detectar y corregir automáticamente no sólo errores tipográficos sencillos, sino también ciertos patrones erróneos de las diferentes grafías prenormativas que todavía conviven con las formas hoy ortográficamente consideradas como correctas.

Por otro lado, la hipótesis de investigación subyacente y más general del mismo es que un corrector ortográfico proporcionaría a los usuarios de la lengua mayor seguridad en la escritura, al tiempo que posibilitaría la familiarización con la ortografía al hablante que la desconozca o la conozca sólo parcialmente (a través de una exposición constante a la misma y el cultivo en este proceso de la memoria visual). Por último, estaríamos dotando a los hablantes de cbk-PH-09 con un recurso lingüístico básico para la escritura de la lengua y la adecuación semiautomática a la ortografía de textos previos a su promulgación.

Por todo ello, el objetivo del trabajo que aquí se presenta ha sido construir un prototipo de corrector ortográfico que cumpla las características mencionadas en la hipótesis de investigación principal, además de habilitar la verificación a largo plazo de la hipótesis de investigación subyacente arriba mencionadas.

La metodología de trabajo ha incluido las siguientes fases: (1) un estudio del estado de la cuestión sobre las aproximaciones y herramientas más comunes para la construcción de correctores ortográficos, así como de los trabajos en detección y clasificación de errores ortográficos del cbk-PH-09 (véase el apartado 2); (2) la selección de la(s) aproximación(es) y herramienta(s) más adecuadas para la construcción del corrector ortográfico de cbk-PH-09 y desarrollo de uno o varios (sub-)sistemas, en caso de haber varias alternativas viables, para su comparación, evaluación, secuenciación y/o integración de los mismos en un único sistema; (3) la implementación y prueba de los distintos módulos y subsistemas considerados en la(s) arquitectura(s) resultante(s); (4) la evaluación y comparación de los sistemas desarrollados, en su caso, para determinar el más adecuado, de acuerdo con las hipótesis de investigación mencionadas.

BY NC ND

4. DESARROLLO DEL CORRECTOR ORTOGRÁFICO PARA EL CBK-PH-09

En este apartado se describirán los aspectos técnicos del desarrollo del corrector ortográfico para el cbk-PH-09. En la búsqueda de una solución óptima para este problema se construyeron finalmente tres subsistemas alternativos diferentes, utilizando (1) Hunspell, (2) un modelo de TAEC y (3) una combinación de Hunspell, el modelo de TAEC y datos del inglés y del tagalo. Cada uno de ellos se presenta a continuación en sendos sub-apartados (4.1, 4.2 y 4.3), ofreciendo en ellos detalles de su diseño, implementación y evaluación propios.

4.1. Desarrollo del sistema basado en Hunspell

Por su gran nivel de implantación, se partió de la premisa de que la aproximación óptima debería basarse o ser combinable con el uso de Hunspell. El objetivo sería usar esta solución de forma directa en caso de ser óptima y, en caso de no ser una solución completamente satisfactoria, procurar contrarrestar sus desventajas al integrarla con otra(s) aproximación(es), aprovechándose al tiempo sus ventajas. De esta manera, se haría posible su uso, además de garantizar la compatibilidad y la interoperabilidad del corrector resultante con otros sistemas ya existentes (navegadores, editores de textos, etc.).

4.1.1. Diseño

En este apartado se describen los componentes esenciales que hubo que desarrollar para construir el corrector basado en Hunspell: su fichero de diccionario, con la extensión '.dic', y el de afijos, con extensión '.aff'. Ambos llevan como nombre el código de la lengua que se desea corregir, según la normativa ISO 649.

El fichero del diccionario consta básicamente de una lista de palabras. Estas palabras pueden ir acompañadas o no de una o más etiquetas (que pueden bien ser morfológicas o bien denotar cualquier otro aspecto de la lengua). Estas etiquetas son útiles a la hora de crear reglas que se apliquen a palabras de una clase gramatical específica o bien a un grupo restringido de palabras (véanse los apartados 4.1.2 y 4.3.2). La lista de palabras debe ser lo más exhaustiva posible, ya que Hunspell marca como incorrectas las palabras que no aparezcan en dicha lista.

El fichero de afijos permite especificar una serie de directivas opcionales que detallan las reglas de derivación, las configuraciones básicas del corrector ortográfico (codificación de caracteres, caracteres relacionados, palabras que se ignoran, ponderación de teclas adyacentes del teclado, uso de guiones, etc.), las reglas de sustitución y el algoritmo fonético, todo ello para la lengua en cuestión. La lista completa de directivas puede consultarse en la documentación de Neméth (2018). Aunque no sea estrictamente necesaria, la creación de reglas de afijación permite limitar la extensión de la lista de palabras, que sin ellas crecería de manera exponencial en ciertas lenguas. Se distinguen dos tipos de afijos: prefijos y sufijos, que se denotan por medio de las directivas *'PFX'* y *'SFX'*, respectivamente.

4.1.2. Implementación

En primer lugar, para construir el diccionario, se extrajeron las palabras y clases gramaticales respectivas incluidas en cinco obras lexicográficas del cbk-PH-09 (Camins 1999; Miravite et al. 2009; Santos, R. A. 2010; Santos 2011; Maletsky 2019),

CC BY NC ND

revisadas y corregidas manualmente para adecuar su escritura a la ortografía. La lista de palabras original constaba de 13.599 palabras con sus etiquetas morfológicas correspondientes, según el etiquetario siguiente:

- *B*: adverbio
- *C*: conjunción
- *D*: determinante
- *I*: interjección
- *J*: adjetivo
- *M*: numeral
- *N*: sustantivo
- *O*: adposición (preposición o locución prepositiva)
- *P*: pronombre
- *R*: abreviatura/acrónimo
- *U*: otros
- *V*: verbo
- *Y*: verbalizador (man/hace)

En segundo lugar, en el fichero de afijos se incluyeron los sufijos y prefijos productivos que poseían un comportamiento relativamente regular. Los afijos implementados fueron:

- *aka-*, *ika-*: prefijos de numeral ordinal
- *ma-*: prefijo adjetivador
- *maka-*: prefijo causativo
- *pagka-*: prefijo abstractivizador
- *ka-*: prefijo de igualdad
- *pinaka-*: prefijo de superlativo absoluto
- *-r*: sufijo verbal infinitivo
- *-han*: sufijo verbal de mutualidad
- *-ando*, *-iendo*: sufijos verbales de gerundio
- *-ao*, *-ido*: sufijos verbales de participio
- *-ada*: sufijo nominalizador de verbos

Una vez eliminadas las redundancias generadas en la lista de palabras por la implementación de los afijos, ésta se redujo a 11.812 palabras. En la lista se mantuvieron todas las locuciones originales, aun cuando todos sus elementos se encontraban en la lista de palabras, ya que su existencia en el diccionario las prioriza como candidatos de corrección en el caso de que éstas se escriban sin espacios.

En tercer lugar, se consideró la posibilidad de usar también el algoritmo de procesamiento fonético de Hunspell. Sin embargo, Hunspell atribuye la mínima prioridad a los candidatos recuperados por esa vía, por lo que el uso del algoritmo fonético generalmente no tiene ningún impacto en el rendimiento del corrector ortográfico. Así, pues, se decidió descartar esta opción en este trabajo.

Finalmente, otra función disponible en Hunspell permite usar reglas de sustitución. No obstante, su codificación requería poseer un conocimiento de la lengua que aún no se había constatado empíricamente, por lo que esta opción no fue usada en esta implementación inicial, sino más adelante, en el sistema híbrido (véase el apartado 4.3).

BY NC ND

4.1.3. Evaluación

En la evaluación de los correctores ortográficos, se han seguido los siguientes convenios y fórmulas para el cálculo de las métricas habituales (tasa de acierto, precisión, cobertura y medida-F):

- Verdadero Positivo (VP): la palabra es correcta y el corrector la detectó como tal.
- Falso Positivo (FP): la palabra es incorrecta y el corrector la detectó como correcta o, habiéndola detectado como incorrecta, no fue capaz de proporcionar el candidato de corrección esperado.
- Verdadero Negativo (VN): la palabra es incorrecta y el corrector la detectó como tal y la corrigió correctamente.
- Falso Negativo (FN): la palabra es incorrecta y el corrector erróneamente la detectó como correcta.

Su cálculo se realiza mediante las siguientes fórmulas:

$$\text{Tasa de acierto} = \frac{VP + VN}{VP + FP + VN + FN}$$

$$\text{Precisión} = \frac{VP}{VP + FP}$$

$$\text{Cobertura} = \frac{VP}{VP + FN}$$

$$\text{Medida} - \text{F} = 2 * \frac{Precisión * Cobertura}{Precisión + Cobertura}$$

La Tabla 1 muestra los valores de estas métricas alcanzados por el corrector ortográfico basado en Hunspell cuando se le piden uno, dos y tres candidatos de corrección para cada error. Como puede observarse, los valores de cobertura no cambian al considerar más candidatos, ya que la capacidad de detección de errores es constante (depende únicamente del diccionario), pero no así la tasa de acierto y la precisión, que aumentan con el número de candidatos considerados.

Tabla 1. Valores de las métricas para el corrector ortográfico basado en Hunspell

N.º candidatos	1	2	3
Tasa de acierto	60,50%	61,85%	63,84%
Precisión	77,51%	79,01%	80,12%
Cobertura	70%	70%	70%
Medida-F	73,56%	74,24%	74,72%

Ahora bien, también se observa que, aunque Hunspell alcanza un rendimiento razonable al corregir errores ortográficos, este valor relativamente bajo de cobertura

CC BY NC ND

indica gran cantidad de falsos positivos. Otro elemento que afecta el rendimiento de Hunspell es la distancia entre las diferentes variantes gráficas actualmente en uso y la forma ortográfica. Las discusiones pertinentes se hacen en el apartado 5.

Este rendimiento de Hunspell, inferior a lo esperado, hizo recomendable probar una aproximación basada en una tecnología alternativa, que pudiera posteriormente combinarse con él y complementarlo en caso necesario. Para ello, un requisito esencial para la elección de esta otra aproximación era que se pudieran extraer y/o interpretar de manera explícita los conocimientos adquiridos con esta aproximación. Así las cosas, se descartaron las aproximaciones no lineales, como es el caso de las redes de neuronas, frente a las que se impuso el sistema alternativo descrito en el subapartado 4.2.

4.2. Desarrollo del sistema basado en Moses (TAEC)

Efectivamente, la TAEC parecía ser una aproximación sumamente adecuada, ya que no sólo parecía una solución prometedora por sí sola para intentar resolver el problema de forma eficiente o, en caso contrario, integrable con la solución basada en Hunspell, pues 1) permitiría acceder a las tablas de frases generadas durante la fase de entrenamiento y 2) posibilitaría una generalización de reglas a partir de la extracción de patrones. De esta manera, se decidió utilizar Moses (véase el subapartado 2.1.3) para entrenar un modelo de TAEC para corregir los errores ortográficos del cbk-PH-09.

4.2.1. Diseño

Para entrenar un modelo de TAEC con Moses era necesario preparar las siguientes entradas: 1) un corpus paralelo bilingüe (o, en nuestro caso, un corpus con textos en grafías no normalizadas en el sentido de la LF y los mismos textos escritos de acuerdo con la ortografía en el sentido de la LM) preprocesado, que puede estar anotado o no, y 2) un modelo lingüístico generado a partir del corpus.

4.2.2. Implementación

En este apartado, se describen los detalles del entrenamiento del modelo de TAEC con Moses.

Por un lado, como corpus paralelo de entrenamiento se usaron aproximadamente 102.000 palabras, extraídas del corpus de cbk-PH-09 escrito *Contemporary Written Zamboangueño Chavacano Corpus* (Himoro y Pareja-Lora, en prensa). Este subcorpus contenía muestras de diferentes géneros, lo que garantiza así el máximo grado de diversidad y cobertura posible. Estas muestras fueron segmentadas (tokenizadas) y corregidas manualmente.

Para que el modelo entrenado utilizara letras como unidad de alineamiento, cada letra fue separada por un espacio, tal como requería esta aproximación. Además, cada palabra fue flanqueada por los caracteres *'#'* y *'$'* (principio y fin de palabra, respectivamente), permitiendo que el algoritmo identifique el contexto en el que ocurren las transformaciones, sea al comienzo, en el medio o al final. La limitación que esto impone, sin embargo, es que se pierde el contexto de la palabra dentro de la frase.

Donde fue necesario utilizar espacios, estos se sustituyeron por el carácter *multi-byte* de espacios (U+3000 en la codificación UTF-8). Dado que Moses utiliza XML internamente, se convirtieron los caracteres especiales a entidades HTML (o texto

BY NC ND

en el caso de los *emojis*) para evitar problemas. Finalmente, las mayúsculas se cambiaron a minúsculas para evitar una dispersión en los datos a la hora de entrenar el modelo. La Figura 4 muestra un fragmento de los datos de entrenamiento utilizados (con grafías prenormativas en la primera columna y de acuerdo con la ortografía en la segunda). Como alineador de palabras, se utilizó GIZA++ (Och y Ney 2003).

```
# a b l a $              # h a b l a $
# y a l a n g $          # y a   l a n g $
# e r a $                # e r a $
# e l $                  # e l $
# d e b e r a s a n $    # d e v e r a s a n $
```

Figura 4. Fragmento de los ficheros de los datos de entrenamiento en ambos sentidos (verso extraído de Pondong 2009)

Por otro lado, como modelo lingüístico se utilizó SRILM (Stolcke, 2002), generado con tetragramas (n = 4, determinado empíricamente en este trabajo).

4.2.3. Evaluación

Para la evaluación de este segundo corrector, se ha utilizado una validación cruzada (*k-fold*) con *k*=10 pliegues, considerando como máximo los 3 primeros candidatos de corrección. La Tabla 2 muestra los valores de las diferentes métricas arrojados por esta validación cruzada por el modelo final de corrector ortográfico basado en TAEC, considerando uno, dos y tres candidatos, respectivamente.

Tabla 2. Valores de las métricas para el corrector ortográfico de TAEC con Moses

N.º candidatos	1	2	3
Tasa de acierto	88,26%	93,6%	94,9%
Precisión	93,01%	96,38%	97,19%
Cobertura	92,22%	95,58%	96,41%
Medida-F	92,61%	95,98%	96,8%

Los valores de la tabla muestran un rendimiento más que aceptable en todas las métricas, y son bastante superiores a los obtenidos para Hunspell, además de crecer con el número de candidatos considerados. Sin embargo, así como ocurre en ese otro caso, esta aproximación también presenta desventajas considerables, al menos cuando se utiliza independientemente. Por ejemplo, debido a las limitaciones del algoritmo de TAEC subyacente y de la manera en la que están estructurados los datos, todos los modelos de TAEC entrenados generan correcciones palabra por palabra, independientemente del contexto global en el que ocurren. Por ello, no son capaces de tratar casos de ambigüedad.

Todo esto se discute en detalle en el apartado 5. El subapartado siguiente trata del desarrollo de un sistema híbrido basado en la implementación hecha con

CC BY NC ND

Hunspell (véase apartado 4.1) con algunas mejoras, derivadas del conocimiento empírico obtenido del desarrollo de la aproximación basada en la TAEC (Moses).

4.3. Desarrollo del sistema híbrido

Con miras a compensar algunas de las deficiencias de Hunspell en lenguas como el cbk-PH-09, cuyas diferentes formas gráficas preexistentes presentan una distancia considerable con sus formas ortográficas (véase el apartado 5), se propuso finalmente una versión mejorada y combinada con el conocimiento obtenido mediante la aproximación anterior, que se detallará en este apartado.

4.3.1. Diseño

Esta nueva aproximación debía partir del corrector ortográfico implementado con Hunspell (véase el apartado 4.1), mejorado mediante una codificación e incorporación adecuadas de dos piezas de conocimiento distintas adicionales:

1. un conjunto de reglas obtenidas de las tablas de traducción del modelo de TAEC (véase el apartado 4.2)
2. un subconjunto de palabras de otras lenguas que a menudo se entremezclan con el cbk-PH-09

Lo que se intentaba probar en este caso es si (1) podría incrementar la capacidad de corrección de la herramienta (y, con ello, los valores de las métricas de tasa de acierto y de precisión), y si (2) podría elevar la capacidad de detección de errores ortográficos (y, por consiguiente, la métrica de cobertura), lo que proporcionaría un incremento de la medida-F.

4.3.2. Implementación

El paso inicial para la implementación de la primera mejora fue la extracción de la tabla de frases de Moses. El proceso es muy sencillo: se trata de extraer de dentro del modelo entrenado el fichero de texto *phrase-table* comprimido con formato GZIP (generado por el modelo basado en TAEC). La Figura 5 muestra un fragmento de dicho fichero.

```
a b ||| h a b ||| 1 0.750703 0.904762 0.0987712 ||| 0-0 1-2 2-3 ||| 19 21 19 ||| |||
k e ||| q u e ||| 0.888889 0.457563 0.242424 0.04582 ||| 0-0 1-1 1-2 2-3 ||| 18 66 16 ||| |||
q u i ||| k i ||| 1 0.00727155 0.428571 0.0871329 ||| 0-0 1-1 3-2 ||| 3 7 3 ||| |||
k ||| c ||| 0.521898 0.557491 0.451104 0.408163 ||| 0-0 ||| 274 317 143 ||| |||
k ||| q u ||| 0.897959 0.597732 0.277603 0.0515475 ||| 0-0 0-1 ||| 98 317 88 ||| |||
s e ||| c e ||| 1 0.168036 0.259259 0.171254 ||| 0-0 1-1 2-2 ||| 14 54 14 ||| |||
```

Figura 5. Fragmento de la tabla de traducciones del modelo basado en TAEC generado por Moses

Cada línea del fichero contiene hasta 7 campos (algunos son opcionales): la frase en la LF, la frase supuestamente equivalente en la LM, las probabilidades de traducción asociadas (traducción inversa, peso léxico inverso, traducción directa, peso léxico directo), los alineamientos entre las dos frases, los valores de rasgos dispersos y otras propiedades clave-valor.

CC BY NC ND

En segundo lugar, debía realizarse la poda de la tabla de frases. Para ello, se utilizó la implementación del método de poda descrito en Johnson, Martin, Foster y Kuhn (2007) incluida en el paquete de Moses. Según estos autores, la poda por relevancia estadística (en inglés, *statistical significance* o *p-value*) parte del principio de que ciertas entradas podrían ser eliminadas de la tabla, ya que 1) la probabilidad de que vuelvan a ocurrir es baja y 2) su aparición en el corpus podría ser marginal y generar simplemente ruido en las traducciones. La poda evita así el sobreajuste del modelo entrenado, es decir, que éste incorpore elementos específicos de los datos de entrenamiento que, además de no mejorar las predicciones del modelo, incluso pueden degradar su rendimiento.

Como parámetros, la herramienta toma un valor umbral *'l'* que, según los autores, debe ser *'α+ε'*[3] para que se eliminen las frases que sólo ocurren una vez en el corpus. Otro parámetro de ajuste es la cantidad máxima *n* de traducciones que deben considerarse para una frase en la LF. En esta implementación se utilizaron los valores $l = \alpha+\varepsilon$ y $n = 5$ (tras determinarse empíricamente este último valor). De esa forma, de las 10.255 frases existentes en la tabla se conservaron únicamente 3.189. Este número de reglas pudo reducirse aún más (1.043, finalmente), por medio de una segunda poda que eliminaba las redundancias: siempre que había un solapamiento entre reglas, se optó por mantener la más general frente a la(s) específica(s).

En tercer lugar, debía convertirse la tabla de frases al formato aceptado por Hunspell, para lo que se eliminaron los espacios entre los caracteres (incluidos a la hora de entrenar el modelo de TAEC), siendo reemplazados por *'_'*, y el carácter *'#'* por *'^'* como identificador de inicio de palabra, y finalmente se agregó la directiva *'REP'* al comienzo de cada línea. La primera línea debía indicar el número de reglas existentes. Luego, tan solo quedaba insertar las reglas resultantes en el fichero de afijos. La Figura 6 muestra un fragmento de las reglas generadas.

```
REP 1043
REP ^ab hab
REP ^ak aqu
REP ^aly all
REP ^al all
REP ^al a_l
REP aw$ ao
```

Figura 6. Ejemplos de reglas de sustitución de la tabla de frases del modelo de TAEC entrenado con Moses convertidas al formato de entrada de Hunspell

Por lo que respecta a la segunda mejora, ésta iba destinada a salvar el problema de los falsos positivos, discutido en detalle en el subapartado 5.2.2. Decidimos, así, incorporar datos de las lenguas que se entremezclan más a menudo con el cbk-PH-09 que poseyeran correctores ortográficos en Hunspell u otro formato compatible (en este caso, el inglés y el tagalo).

En relación con el léxico, se incluyó la totalidad de las palabras de los diccionarios de las dos lenguas bajo la etiqueta *'!'*. Para evitar que éstas se ofrecieran como candidatos de corrección, se aplicó la directiva *'NOSUGGEST'* a todas las palabras que llevaran dicha etiqueta. En cuanto a los afijos, cabe destacar que, pese a la gran cantidad de prefijos, infijos y sufijos utilizados en tagalo, la implementación

[3] Como estas entradas tienen el valor p más bajo (1/N según la prueba exacta de Fisher), con el valor umbral α+ε (ε siendo un número positivo muy pequeño) se eliminan todas ellas.

CC BY NC ND

disponible no incluía ninguna regla de derivación de este tipo, lo que, en cierta medida, limitará la capacidad de nuestro corrector del cbk-PH-09 para detectar palabras de esta lengua. Para el inglés, se incluyeron las reglas de derivación con pequeños ajustes en el nombre de las etiquetas, ya que se solapaban con las usadas para el cbk-PH-09. De esa forma, *a priori*, Hunspell dejaría de señalar como error ortográfico gran parte de las palabras de estas dos lenguas.

4.3.3. Evaluación

A continuación, se ofrecen los valores de las cuatro métricas habituales (tasa de acierto, precisión, cobertura y medida-F) alcanzados por las dos mejoras propuestas para uno, dos y tres candidatos primero por separado (véanse las tablas 3 y 4), y luego conjuntamente (véase la Tabla 5).

Tabla 3. Valores de las métricas para el corrector ortográfico basado en Hunspell tras incorporar únicamente la tabla de traducciones del modelo de TAEC

N.º candidatos	1	2	3
Tasa de acierto	68,08%	69,73%	69,87%
Precisión	86,91%	89,24%	89,45%
Cobertura	69,87%	69,87%	69,87%
Medida-F	77,46%	78,37%	78,38%

Tabla 4. Valores de las métricas para el corrector ortográfico basado en Hunspell tras incorporar únicamente los datos del inglés y del tagalo

N.º candidatos	1	2	3
Tasa de acierto	71,76%	72,81%	73,74%
Precisión	79,76%	80,77%	81,67%
Cobertura	85,81%	85,81%	85,81%
Medida-F	82,67%	83,21%	83,69%

BY NC ND

Tabla 5. Valores de las métricas para el corrector ortográfico basado en Hunspell tras incorporar las tablas de traducciones del modelo de TAEC y los datos del inglés y del tagalo

N.º candidatos	1	2	3
Tasa de acierto	78,03%	79,53%	79,67%
Precisión	86,19%	87,88%	88,03%
Cobertura	85,76%	85,76%	85,76%
Medida-F	85,97%	86,8%	86,88%

5. DISCUSIÓN

En este apartado, se discuten en detalle las ventajas y desventajas tanto de Hunspell como del modelo de TAEC (apartados 5.1 a 5.4). La aproximación híbrida, que hereda parte de las ventajas e inconvenientes de ambas, se discute por separado, en el apartado 5.5.

5.1. Ventajas de Hunspell y desventajas de la TAEC

Como muestra la Tabla 6, ambas aproximaciones son en gran medida complementarias, ya que muchas de las ventajas de Hunspell son desventajas de la TAEC. Aquí se discute brevemente cada una de ellas.

5.1.1. Ampliación de la cobertura (ref. VH1/DT1)

Una de las ventajas más notables de Hunspell es la simplicidad con la que es posible introducir nuevas palabras, pues basta con agregarlas al diccionario. En el caso de la TAEC, es necesario ampliar el conjunto de datos de entrenamiento con nuevos textos y volver a entrenar el modelo. Así pues, los costes de tiempo y humanos son muy superiores en este último caso.

5.1.2. Validez de los candidatos (ref. VH2/DT2)

Si por un lado la dependencia de Hunspell de su diccionario de entrada es una limitación en ciertos aspectos (véase DH2), por otro lado, es su fortaleza, ya que nunca se ofrecerán como candidatos de corrección palabras inválidas o inexistentes. Lo mismo se puede decir para el modelo de TAEC, cuya capacidad de generalizar las reglas aprendidas y generar correcciones para palabras desconocidas es una ventaja en algunos casos (véase VT3), pero puede acabar generando palabras que no existen en la lengua. Esta desventaja del modelo de TAEC trae, además, una implicación adicional que se discute en el apartado 5.1.3.

Sin embargo, una implicación de esta limitación de Hunspell es que, según como se escriba una forma incorrecta, ésta nunca se corregirá realmente. Un ejemplo es el de la palabra 'ojala' ('ojalá' en castellano), que a menudo se escribe como dos palabras: o hala. Puesto que ambas palabras existen en cbk-PH-09 y constan en el diccionario

CC BY NC ND

de Hunspell, el corrector nunca será capaz de corregirla. A fecha de hoy, no hay manera de modificar el comportamiento por defecto de Hunspell.

Tabla 6. Resumen de las ventajas y desventajas de Hunspell y TAEC

	hunspell	TA Estadística de Caracteres
VENTAJAS	▪ **(VH1)** Facilidad de extensión de los diccionarios ▪ **(VH2)** No genera candidatos de corrección inexistentes o inválidos ▪ **(VH3)** Mejor rendimiento con errores aleatorios ▪ **(VH4)** Escalable y respuesta en tiempo real ▪ **(VH5)** Alto grado de interoperabilidad con otros programas y aplicaciones ▪ **(VH6)** Mejor rendimiento con errores específicos/poco frecuentes	▪ **(VT1)** Errores frecuentes aprendidos del corpus ▪ **(VT2)** Capaz de tratar palabras de otras lenguas ▪ **(VT3)** Capaz de generar candidatos de corrección para palabras desconocidas ▪ **(VT4)** Rendimiento numéricamente muy superior al de *hunspell*
DESVENTAJAS	▪ **(DH1)** No corrige de manera adecuada muchos errores frecuentes ▪ **(DH2)** Depende de un diccionario ▪ **(DH3)** Candidatos de corrección limitados a las palabras del diccionario ▪ **(DH4)** Función de algoritmo fonético poco útil e infrautilizada. ▪ **(DH5)** No considera el contexto	▪ **(DT1)** Costes de inclusión de nuevas palabras ▪ **(DT2)** Genera palabras inexistentes o deforma palabras correctas ▪ **(DT3)** Deforma palabras con errores aleatorios ▪ **(DT4)** Lento y no escalable ▪ **(DT5)** Dificultades de integración con otros sistemas computacionales ▪ **(DT6)** No considera el contexto

5.1.3. Corrección de errores aleatorios (ref. VH3/DT3)

Hunspell tiene un rendimiento bastante superior al del modelo de TAEC propuesto para este tipo de palabras. Esto se debe a que el modelo es dependiente de los datos de entrenamiento. Un error de esta naturaleza en muchos casos tiene muy baja probabilidad de volver a producirse y termina por agregar ruido al modelo de TAEC, lo que provoca deformaciones en algunas palabras.

5.1.4. Escalabilidad y tiempo de respuesta (ref. VH4/DT4)

Hunspell, al ser un corrector ortográfico tradicional, es escalable y responde en tiempo real. Con el modelo de TAEC, emular su comportamiento no resulta sencillo, ya que no fue concebido para tal propósito. Además de requerir un tiempo de comunicación con el servidor relativamente elevado, necesita más recursos computacionales para su ejecución. La solución práctica sería revisar el texto completo de una sola vez al final del proceso de redacción, pero con un período de latencia (espera) proporcional a su número de palabras.

CC BY NC ND

5.1.5. Interoperabilidad (ref. VH5/DT5)

Hunspell, ya presente en diferentes plataformas y sistemas, tiene un grado de interoperabilidad elevado, lo que ahorra esfuerzos de integración o necesidades de desarrollo adicional de *software*. Por el contrario, con el modelo de TAEC, es necesario configurar el entorno en el que se ejecutará el servidor – que puede ser local o remoto –, tomar decisiones sobre cómo preprocesar el texto, realizar las llamadas a una API, recibir y procesar la respuesta, e integrarla de manera conveniente para el usuario.

5.2. Debilidades de Hunspell y fortalezas de la TAEC

De igual manera, la Tabla 6 muestra que ambas aproximaciones son en gran medida complementarias, ya que muchas de las ventajas de la TAEC también son desventajas de Hunspell. A continuación, se discute brevemente cada una de ellas.

5.2.1. Corrección de errores frecuentes (ref. DH1/VT1)

Una desventaja de Hunspell en la corrección ortográfica de una lengua con palabras cuyas variantes gráficas difieren bastante de su forma ortográfica es que esta herramienta no siempre alcanza un rendimiento adecuado. En estos casos, la aplicación de un algoritmo fonético podría ser útil para reducir el impacto de este problema; pero, como se discute en el apartado 6.4.1 (ref. DH4), su comportamiento en Hunspell hace que no ofrezca los beneficios esperados y sea por tanto infrautilizado. En estos casos, el modelo de TAEC tiende a obtener mejores resultados, ya que aprende de los datos de entrenamiento y de la frecuencia de ocurrencia en los mismos. La Tabla 7 muestra algunos de los casos en los que Hunspell no se comporta de la manera esperada.

Tabla 7. Errores frecuentes corregidos erróneamente por Hunspell

Tipo de error	Token (forma ortográfica)	Distancia
Omisión del espacio,	dila/diila (de ila)	3/2
Uso de grafemas homófonos	dikaw (de ikaw)	3
	nway (no hay)	3
	kunakel (con aquel)	5
Uso de grafemas homófonos	aki (aqui)	2
	kwando (cuando)	2
	serka (cerca)	2
Interferencia por cognados de otras lenguas, Uso de grafemas homófonos	attaka (ataca)	2

5.2.2. Extranjerismos, préstamos y cambio y mezcla de código (ref. DH2/VT2)

En cbk-PH-09, los extranjerismos y préstamos lingüísticos son extremamente habituales. En contextos formales escritos, los fenómenos de cambio de código (*code switching*) y mezcla de código (*code mixing*) son también frecuentes. Esto tiene una implicación directa en el funcionamiento de Hunspell, que depende de un diccionario y, por lo tanto, señala como incorrectas todas las palabras que no consten en éste, además de intentar encontrar para ellas un posible candidato de corrección. Esto degrada bastante su rendimiento, ya que los casos de falsos positivos son bastante elevados. El modelo de TAEC, por su parte, es capaz de reconocer y aprender las

CC BY NC ND

palabras que aparezcan con frecuencia en los datos de entrenamiento, evitando correcciones innecesarias y los casos de falsos positivos más típicos.

5.2.3. Corrección de palabras desconocidas (ref. DH3/VT3)

Como ya se ha mencionado anteriormente, Hunspell no es capaz de generar correcciones para palabras desconocidas o que no consten en su diccionario. El modelo de TAEC, por otro lado, por medio de la generalización de patrones aprendidos en la fase de entrenamiento, puede generar correcciones aun cuando encuentre una palabra por primera vez. Si por un lado es una desventaja de Hunspell y una ventaja del modelo de TAEC en algunas situaciones, también puede tener el efecto contrario (véase el apartado 5.1.2, ref. VH2/DT2).

5.3. Desventajas comunes de Hunspell y el modelo de TAEC (ref. DH5/DT6)

Una desventaja compartida por ambas aproximaciones es que ninguna de las dos tiene en consideración el contexto. En el caso de Hunspell, por ejemplo, si una palabra se encuentra en el diccionario, siempre será aceptada como correcta. En el caso del modelo de TAEC, puesto que el contexto de la frase es inexistente, siempre se seleccionará la entrada de mayor probabilidad en la tabla de frases. En ambos casos, el comportamiento es fijo, ya que la corrección se hace palabra a palabra: siempre la marcará como incorrecta y la corregirá de una cierta manera, o bien la aceptará como correcta.

5.4. Aspectos exclusivos

5.4.1. Problemas con los algoritmos fonéticos en Hunspell (DH4)

Como ya se ha mencionado en el apartado 5.2.1, el uso del algoritmo fonético de Hunspell es bastante limitado, ya que los candidatos por él generados tienen una prioridad mínima. Sería conveniente que hubiera una manera de cambiar su funcionamiento y aumentar su prioridad por encima de otras formas de generación de candidatos de corrección mediante alguna directiva específica, pero de momento esta opción no existe.

5.4.2. Rendimiento superior de la TAEC

Como se ha visto previamente, el rendimiento del modelo de TAEC es superior al de Hunspell. Sin embargo, éste también presenta desventajas importantes que no se pueden obviar (v.g., DT1, DT2 y DT3).

5.5. Evaluación cualitativa de la versión híbrida del corrector ortográfico en Hunspell

La versión híbrida del corrector ortográfico conserva todas las ventajas de Hunspell, ya que también está basada en él. Además, incorpora parcialmente la ventaja del modelo de TAEC de incorporar reglas que cubren los errores más frecuentes (ref. VT1), mitigando así el problema de errores frecuentes que no se corregían adecuadamente con Hunspell (ref. DH1), pero con la ventaja de que no genera candidatos inválidos, como puede llegar a pasar con el modelo de TAEC. La

CC BY NC ND

Tabla 8 resume las desventajas de Hunspell mitigadas y las ventajas del modelo de TAEC parcialmente incorporadas en la versión híbrida.

Tabla 8. Desventajas de Hunspell mitigadas y ventajas del modelo de TAEC parcialmente incorporadas en la versión híbrida de Hunspell (leyenda: ✓ presente, ½ parcialmente presente, × no presente)

Característica	Estado
DH1	½
DH2	½
DH3	✓
DH4	✓
DH5	✓
VT1	½
VT2	½
VT3	×
VT4	½

Desde el punto de vista del rendimiento, en relación con la implementación inicial en Hunspell, se observa un incremento medio de un 17,01% en la tasa de acierto, un 8,49% en la precisión, un 15,76% en la cobertura y un 12,38% en la medida-F. No obstante, no hay que olvidar que los valores obtenidos aún son inferiores a los obtenidos con el modelo de TAEC.

6. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo, se muestra primeramente el desarrollo de dos correctores ortográficos para el chabacano de Zamboanga: uno utilizando Hunspell, la herramienta más usada para la corrección ortográfica, y otro utilizando un modelo de Traducción Automática Estadística de Caracteres (TAEC), entrenado con datos de un corpus, corregidos manualmente. Su construcción ha demostrado que ambas aproximaciones tienen ventajas y desventajas casi complementarias, y que, por lo tanto, resultaba interesante encontrar una manera de combinarlas de alguna manera. En consecuencia, se optó por crear un nuevo sistema mediante la integración del modelo de TAEC en Hunspell, una vez que se juzgó primordial garantizar la usabilidad y la interoperabilidad con otros sistemas existentes.

Este nuevo sistema incorporaba dos mejoras al corrector basado únicamente en Hunspell, con el fin de compensar algunas de sus debilidades: 1) la introducción de la tabla de frases del modelo de TAEC como reglas de sustitución, combinando así ambas aproximaciones, y 2) la introducción de datos del inglés y de tagalo. Al combinar ambas mejoras, se ha obtenido un corrector híbrido en un doble sentido: se combinan reglas obtenidas por medio de una técnica de aprendizaje automático con reglas codificadas manualmente, y se combinan los léxicos de dos de las lenguas con las que el chabacano tiende a mezclarse más hoy en día. Con la implementación de esta aproximación se ha observado un incremento en los valores de todas las métricas con respecto al sistema basado únicamente en Hunspell y, aunque su rendimiento todavía esté por debajo del que proporciona la aproximación basada en TAEC, se ha logrado un rendimiento aceptable, a la par que se conserva la interoperabilidad y la escalabilidad proporcionada por Hunspell.

Como continuación de este estudio, proponemos revisar completamente la lista de palabras del diccionario de Hunspell, incorporar al mismo vocablos de otras dos obras

CC BY NC ND

lexicográficas no incluidas anteriormente (Ariston, 2002; Chambers, 2003) y adecuar la grafía utilizada con las formas que constan en el diccionario oficial de chabacano zamboangueño, publicado por el gobierno local. Una vez completadas estas tareas, sería ya posible hacer pública una primera versión del corrector.

REFERENCIAS

Ariston, E. M. 2002. *Chabacano-English dictionary*. Manila: Ms.

Atkinson, K. 2019. *Aspell* (Vers. 0.60.8). <http://aspell.net/> (acceso: 19/06/2021)

Avanço, L. V., Sanches Duran, M. y Volpe Nunes, M. G. 2014. Towards a honetic Brazilian Portuguese spell checker. *International Conference on Computational Processing of the Portuguese Language – PROPOR*. Porto Alegre: Sociedade Brasileira de Computação, 24–31. <https://repositorio.usp.br/item/002516552> (acceso: 20/09/2021)

Beinborn, L., Zesch, T. y Gurevych, I. 2013. Cognate production using character-based machine rranslation. *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Nagoya: Asian Federation of Natural Language Processing, 883–891. <https://www.aclweb.org/anthology/I13-1112/> (acceso: 03/02/2019)

Bernhard, D. y Steiblé, L. 2015. Quand l'oral se fait entendre à l'écrit: alignement de lexiques en l'absence de normalisation graphique. *TALaRE 2015 - Traitement Automatique des Langues Régionales de France et d'Europe*. <https://hal.archives-ouvertes.fr/hal-01158489> (acceso: 20/09/2021)

Bhatti, Z., Waqas, A., Ismaili, I. A., Hakro, D. N. y Soomro, W. J. 2014. Phonetic based soundex & shapeex algorithm for sindhi spell checker system. *Advances in Environmental Biology* 8(4): 1147–1155. < https://arxiv.org/abs/1405.3033> (acceso: 15/06/2021)

Camins, B. S. 1999. *Chabacano de Zamboanga handbook and Chabacano-English-Spanish dictionary* (2ª ed.). Ciudad de Quezón: Claretian Publications.

Chambers, J. 2003. *English-Chabacano dictionary*. Ciudad de Zamboanga: Ateneo de Zamboanga University Press.

Damerau, F. J. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM* 7(3): 171–176.

DepEd Zamboanga City Division. 2016. *Revised Zamboanga Chavacano orthography (Guía para la enseñanza de Chavacano)*. Zamboanga: Zamboanga City Local Government.

Etoori, P., Chinnakotla, M. y Mamidi, R. 2018. Automatic spelling correction for resource-scarce languages using deep learning. *Proceedings of ACL 2018, Student Research Workshop*. Melbourne: Association for Computational Linguistics, 146–152. <https://aclanthology.org/P18-3021/> (acceso: 16/06/2021)

Gao, J., Li, X., Micol, D., Quirk, C. y Sun, X. 2010. A large scale ranker-based system for search query spelling correction. *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*. Association for Computational Linguistics, 358–366.

Ghosh, S. y Kristensson, P. O. 2017. *Neural networks for text correction and completion in keyboard decoding*. <https://arxiv.org/abs/1709.06429> (acceso: 16/06/2021)

Gorin, R. E. 1974. *SPELL: Spelling Check and Correction Program*. <https://www.saildart.org/allow/SPELL.REG%5BUP,DOC%5D> (acceso: 01/09/2019)

Government of the Philippines. 2011. DepEd develops learning supplements using mother-tongue. *Official Gazette*. <https://www.officialgazette.gov.ph/2011/11/28/deped-develops-learning-supplements-using-mother-tongue/> (acceso: 19/01/2017)

Han, B., P. Cook and Baldwin, T. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology* 4(1): 1–27.

Hasan, S., C. Heger y Mansour, S. 2015. Spelling correction of user search queries through statistical machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon: Association for Computational Linguistics, 451–460. <https://www.aclweb.org/anthology/D15-1051/> (acceso: 03/02/2019)

CC BY NC ND

Hendricks, K. 2011. *MySpell* (Vers. 3.0) [Software]. <https://code.google.com/archive/a/apache-extras.org/p/ooo-myspell> (acceso: 01/09/2019)

Himoro, M. Y. 2019. *Hacia un corrector ortográfico para la nueva ortografía del chabacano de Zamboanga*. Trabajo de fin de máster, Universidad Nacional de Educación a Distancia. <http://e-spacio.uned.es/fez/view/bibliuned:master-Filologia-TICETL-Myhimoro> (acceso: 16/06/2021).

Himoro, M. Y. y Pareja-Lora, A. 2020. Towards a spell checker for Zamboanga Chavacano orthography. *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, 2685–2697. <https://aclanthology.org/2020.lrec-1.327/> (acceso: 16/06/2021)

Himoro, M. Y. y Pareja-Lora, A. En prensa. Construcción de un corpus escrito y una ontología de errores ortográficos del Chabacano Zamboangueño. *Revista Española de Lingüística Aplicada/Spanish Journal of Applied Linguistics*.

Ikeda, T, Shindo, H. y Matsumoto, Y. 2016. Japanese text normalization with encoder-decoder model. *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, 129–137. <https://aclanthology.org/W16-3918/> (acceso: 16/06/2021)

Karimi, S. 2008. *Machine transliteration of proper names between English and Persian*. Tesis doctoral, RMIT University. <https://researchrepository.rmit.edu.au/esploro/outputs/doctoral/Machine-transliteration-of-proper-names-between-English-and-Persian/9921864000401341> (acceso: 03/02/2019).

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A. y Herbst, E. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Prague: Association for Computational Linguistics, 177–180.

Korchagina, N. 2017. Normalizing Medieval German texts: From rules to deep learning. *NoDaLiDa 2017 Workshop on Processing Historical Language*. Gothenburg: Linköping University Electronic Press, 12–17. <https://www.aclweb.org/anthology/W17-0504/> (acceso: 03/02/2019)

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10(8): 707–710.

Maletsky, F. H. 2019. *Chavacano-Chabacano: The original online Chavacano-English dictionary*. <https://www.zamboanga.com/chavacano/chavacano_de_zamboanga_speak.htm> (acceso: 12/01/2019)

Miravite, R. M., Sanchez, U. C. N., Tardo, D. S., Viloria, S. J. B. y Delos Reyes, D. J. M. 2009. *Chavacano reader*. Hyattsville: Dunwoody Press.

Nakov, P. y Tiedemann, J. 2012. Combining word-level and character-level models for machine translation between closely-related languages. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* 2. Jeju Island: Association for Computational Linguistics, 301–305.

National Statistics Office. 2014. 2010. *Census of population and housing. Report No. 2B – Population and household characteristics (Sample variables)*. Manila: National Statistics Office. <https://psa.gov.ph/sites/default/files/2010 CPH Report No. 2B - PHILIPPINES.pdf> (acceso: 19/01/2017)

Németh, L. 2016. *Hunspell's website*. <http://Hunspell.github.io/> (acceso: 19/01/2017)

Németh, L. 2018. *Hunspell* (Vers. 1.7.0) [Software] (acceso: 19/06/2021).

Németh, L., Trón, V., Halácsy, P., Kornai, A., Rung, A. y Szakadát, I. 2004. Leveraging the open source Ispell codebase for minority language analysis. *Proceedings of SALTMIL Workshop at LREC 2004: First Steps in Language Documentation for Minority Languages*. Reykjavik: European Language Resources Association, 56–59. <http://ixa2.si.ehu.eus/~jipsagak/SALTMIL/LREC_2014_Workshop_Proceedings_Saltmil.pdf> (acceso: 03/02/2019)

Och, F. J. y Ney, H. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1): 19–51.

Philips, L. 2000. The double metaphone search algorithm. *C/C++ Users Journal* 18: 38–43.

CC BY NC ND

Poibeau, T. 2017. *Machine translation.* Cambridge: MIT Press.
Pondong. 2009. *Promesa.* <https://www.youtube.com/watch?v=rmWLXE_DTng> (acceso: 05/10/2017)
Russell, R. C. 1918. Soundex. US1261167.
Saint-Exupéry, A. 1943. Le Petit Prince. *Ebooks libres et gratuits.* <https://www.cmls.polytechnique.fr/perso/tringali/documents/st_exupery_le_petit_prince.pdf> (acceso: 21/02/2020)
Saint-Exupéry, A. 2018. *El Diutay Principe.* Traducido por Jerome Herrera. Manila: Jerome Herrera.
Santos, F. M. 2011. *Chavacano handbook: El español que se habla en Zamboanga. Usage and dictionary.* Ciudad de Zamboanga: Linus Multi-Media & Editorial Services.
Santos, R. A. 2010. *Chavacano de Zamboanga: Compendio y diccionario Chavacano-English / English-Chavacano.* Ciudad de Zamboanga: Ateneo de Zamboanga University Press.
Scherrer, Y., T. Samardžić y Glaser, E. 2016. Normalizing orthographic and dialectal variants in the ArchiMob corpus of spoken Swiss German. Presentado en '6th Days of Swiss Linguistics'. Ginebra, 29 de junio – 01 de julio. <https://archive- ouverte.unige.ch/unige:90850> (acceso: 03/02/2019)
Schneider, G., Pettersson, E. y Percillier, M. 2017. Comparing rule-based and SMT-based spelling normalisation for English historical texts. *NoDaLiDa 2017 Workshop on Processing Historical Language*. Gothenburg: Linköping University Electronic Press, 40–46. <https://www.aclweb.org/anthology/W17-0508/> (acceso: 03/02/2019)
Stolcke, A. 2002. SRILM – An extensible language modeling toolkit. *Proceedings of the 7th International Conference on Spoken Language Processing [ICSLP 2002].* Denver: ISCA, 901–904. <https://www.isca-speech.org/archive/icslp_2002/i02_0901.html> (acceso: 16/06/2021)
Tiedemann, J. 2009. Character-based PSMT for closely related languages. *Proceedings of the 13th Annual Conference of the EAMT*. Barcelona: European Association for Machine Translation, 12–19.
Tiedemann, J. y Nabende, P. 2009. Translating transliterations. *International Journal of Computing and ICT Research* 3: 33–41.
Tursun, O. y Cakici, R. 2017. Noisy Uyghur text normalization. *Proceedings of the 3rd Workshop on Noisy User-generated Text*, 85–93. <https://aclanthology.org/W17-4412> (acceso: 16/06/2021)
UzZaman N. y Khan, M. 2005. A double metaphone encoding for Bangla and its application in spelling checker. *International Conference on Natural Language Processing and Knowledge Engineering*. Wuhan: Institute of Electrical and Electronics Engineers, 705–710.
Vilar, D., Peter, J.-T. y Ney, H. 2007. Can we translate letters? *Proceedings of the Second Workshop on Statistical Machine Translation*. Prague: Association for Computational Linguistics, 33–39.
Willisson, P. 2021. *Ispell. GNU Operating System*. <https://www.gnu.org/software/ispell> (acceso: 16/06/2021)
Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D. y Ng, A. Y. 2016. *Neural language correction with character-based attention*. <https://arxiv.org/abs/1603.09727> (acceso: 18/06/2021)

CC BY NC ND