ISSN 1870-4069

Corrección de errores del reconocedor de voz de Google usando métricas de distancia fonética

Diego Campos Sobrino[1], Mario Campos Soberanis[1], Iván Martínez Chin[1,2], Víctor Uc Cetina[1,2]

[1] SoldAI Research, Mérida, México

[2] Universidad Autónoma de Yucatán, Facultad de Matemáticas, Mérida, México

{dcampos,mcampos}@soldai.com, imartinezchin@gmail.com, uccetina@correo.uady.mx

Resumen Los errores en los sistemas de reconocimiento de voz para el idioma español, como por ejemplo el de Google, ocurren con bastante frecuencia cuando se utilizan en aplicaciones de un dominio específico. Estos errores se presentan mayormente cuando se intenta reconocer palabras que son nuevas para el modelo de lenguaje del reconocedor y que son *ad hoc* al dominio. En este artículo se presenta un algoritmo que usa la distancia de Levenshtein sobre fonemas para reducir el error del reconocedor de voz. Los resultados preliminares muestran que es posible corregir los errores del reconocedor de manera importante mediante el empleo de esta métrica y el uso de un diccionario de frases específicas del dominio de la aplicación. El algoritmo que aquí se propone, a pesar de estar diseñado para dominios muy específicos, es de aplicación general. Es decir, las frases que deben ser reconocidas pueden ser definidas específicamente para cada aplicación, sin que el algoritmo deba modificarse. Basta con indicarle al algoritmo el conjunto de frases sobre las cuales debe trabajar. La complejidad del algoritmo es $O(tn)$, donde t es el número de palabras contenidas en la transcripción que se requiere corregir y n es el número de frases específicas del dominio.

Palabras clave: reconocedor de voz, Levenshtein, corrector fonético.

Fixing Errors of the Google Voice Recognizer through Phonetic Distance Metrics

Abstract. The errors in speech recognition systems for Spanish language such as Google occur quite frequently when used in applications of a specific domain. These errors occur mostly when trying to recognize words that are new to the recognizer's language model and that are *ad hoc* to the domain. In this article we present an algorithm that uses Levenshtein distance on phonemes to reduce the error of the speech recognizer. The preliminary results show that it is possible to correct the errors of the recognizer in an important way by using this metric and the use of a

dictionary of specific phrases from the domain of the application. The algorithm proposed here, despite being designed for very specific domains, is of general application. That is, the phrases that must be recognized can be defined specifically for each application, without the algorithm having to be modified. It is enough to indicate to the algorithm the set of sentences on which it must work. The complexity of the algorithm is $O(tn)$, where t is the number of words contained in the transcript to be corrected and n is the number of phrases specific of the domain.

Keywords: voice recognizer, Levenshtein, phonetic corrector.

1. Introducción

Tradicionalmente los algoritmos utilizados para transformar audio a texto han sido diseñados usando modelos probabilísticos, como los modelos ocultos de Markov. Sin embargo, con el resurgimiento de la redes neuronales, también se están desarrollando redes neuronales profundas [3], lo que ha permitido generar reconocedores de voz más precisos. Ahora bien, cuando estos reconocedores se utilizan en dominios muy específicos, es de esperarse que su error se incremente. Un dominio muy específico, en el contexto de este artículo, hace referencia a un problema de reconocimiento de voz en donde además de reconocer las palabras de un modelo general del idioma, también se requiere reconocer un conjunto de frases que contienen palabras que no existen en ese modelo general, ya que son palabras creadas y con significado exclusivamente dentro de una aplicación particular. Un caso muy claro donde se dan estos dominios muy específicos son los restaurantes, donde es común poner nombres llamativos a sus platillos o promociones. Por ejemplo en la frase "jueves mozzaleroso", "mozzareloso" es una palabra creada por el dueño de una pizzería, y es prácticamente imposible que sea reconocida por un reconocedor de voz para el idioma español. Para aliviar este problema el reconocedor de Google ofrece la opción de especificar una lista de frases específicas de nuestro dominio de aplicación con la finalidad de hacerlas más probables en su modelo de lenguaje, y como resultado tengan mayor probabilidad de ser seleccionadas como la palabra reconocida. Sin embargo, en la práctica esta estrategia no es tan efectiva y es muy fácil encontrar ejemplos en los cuales no funciona como se esperaría. Es importante hacer énfasis en que el algoritmo que aquí se propone, a pesar de estar diseñado para dominios muy específicos, es de aplicación general. Es decir, las frases que deben ser reconocidas pueden ser definidas específicamente para cada aplicación, sin que el algoritmo deba modificarse. Basta con indicarle al algoritmo el conjunto de frases sobre las cuales debe trabajar.

La forma más común de resolver los errores de reconocimiento de voz es desarrollando un módulo de post-procesamiento de las sentencias reconocidas. En [10] dicho módulo consiste en dos pasos. Primeramente, se genera un espacio de búsqueda conteniendo las palabras alternativas para los errores detectados y posteriormente a las palabras candidatas se les asigna un índice con el cual se decide el reemplazo de palabras más adecuado. Para asignar dicho índice

se utiliza un modelo estadístico de palabras que considera información tanto sintáctica como semántica, basado en las co-ocurrencias de las palabras. Otros investigadores han usado la *Distancia de Relevancia Normalizada* (DRN, o NRD por las siglas en inglés de Normalized Relevance Distance) para corregir estos errores [6]. Dicho método se basa en medir la similitud semántica entre palabras y ha demostrado tener una alta efectividad al usarse en tareas de reconocimiento continuo de voz. Con la NRD es posible identificar no sólo las co-ocurrencias sino hasta la correlación de importancia de los términos en el documento, incluso cuando las palabras están distantes una de otra. También se han propuesto métodos más prácticos, como el uso del corrector ortográfico de Bing [2].

Una propuesta de corrección de errores de reconocimiento basada en n-gramas e información contextual lejana es usada en [11]. El método consiste en dos fases de corrección, inicialmente usando rasgos basados en n-gramas y en un segundo paso aplicando la corrección contextual mediante análisis semántico latente.

En [1] se utiliza un mecanismo de desarrollo evolutivo considerando una sentencia erróneamente reconocida como cigoto y haciéndola crecer con respecto a los genotipos propios del dominio de aplicación. A partir de ahí los fenotipos llenan los espacios vacíos en la sentencia con las palabras específicas del dominio.

Por su parte, [8] propone el uso de un modelo de regresión logística para clasificar alternativas de corrección de texto en una interfaz de reconocimiento de voz, lo cual afirman puede reducir el número de posibles correcciones.

En general, durante los últimos años se han propuesto variedad de métodos para reducir el error de los reconocedores de voz. Una revisión de las técnicas usadas recientemente se hace en [4], donde también se cuestiona la efectividad de las métricas tradicionales de evaluación de dichos sistemas, sin embargo para ello no se considera la similitud fonética para la corrección, ni para la evaluación de los resultados.

Otra métrica alternativa para la evaluación de sistemas de reconocimiento de voz propuesta en [5] considera la interpretación de la frase reconocida por parte de un humano, lo cual no necesariamente resulta de utilidad cuando el destino del reconocimiento de voz es el ulterior procesamiento del lenguaje por un algoritmo.

En este artículo se presenta un algoritmo para corregir los errores del reconocedor de voz de Google para el lenguaje español. Dicho algoritmo está diseñado para aplicaciones donde las palabras que se requieren reconocer son de un dominio muy específico y por lo tanto el modelo de lenguaje general que utiliza el reconocedor de Google presenta errores en su reconocimiento.

El resto del artículo está estructurado de la siguiente manera. En la Sección 2 se describe formalmente el problema de corrección de transcripciones de audio; en la Sección 3 se presenta el algoritmo y se analiza su complejidad en tiempo de cómputo; en la Sección 4 se describe el trabajo experimental realizado con la base de datos de una aplicación de restaurantes; finalmente en la Sección 5 se proporcionan las conclusiones junto con algunas ideas para desarrollar como trabajo futuro.

2. Definición del problema

Dada una transcripción de audio T de m palabras, donde n de esas m palabras fueron incorrectamente reconocidas, se requiere corregir los errores de reconocimiento mediante un algoritmo que sea suficientemente rápido para ser usado en tiempo real.

Comúnmente se pueden encontrar cuatro tipos diferentes de errores en el reconocimiento de las palabras que conforman una frase:

1. Sustitución. Cuando una palabra individual es incorrectamente reconocida y sustituida por otra (ej. "pistas" en lugar de "pizzas").
2. Unión de palabras. Cuando dos o más palabras contiguas son reconocidas como una sola (ej. "proceso" en lugar de "por eso").
3. Separación de palabras. Cuando una palabra es reconocida como dos o más palabras en secuencia (ej. "chile ta" en lugar de "chuleta").
4. División incorrecta. Cuando la separación entre dos palabras se ubica en el fonema incorrecto (ej. "pizarra García" en lugar de 'pizza ragazza").

3. Algoritmo

El procedimiento de corrección de transcripciones propuesto en el Algoritmo 1, toma como entrada una transcripción de audio producida por el sistema de reconocimiento de voz (Speech-to-text o STT por sus siglas en inglés) y un contexto compuesto por un conjunto de frases de una o más palabras *ad hoc* al dominio sobre el cual se realiza el reconocimiento; con estos elementos el algoritmo verifica si existe similitud fonética entre las palabras reconocidas en la frase de entrada y las proporcionadas en el contexto. Para verificar la similitud fonética, el algoritmo transforma la transcripción del sistema STT y las frases del contexto a su representación fonética en el sistema IPA (International Phonetic Alphabet), analiza que segmentos de la transcripción son susceptibles de ser corregidos y calcula su similitud con los elementos del contexto, eligiendo las frases contextuales con mayor similitud como frases candidatas. Estas frases candidatas son consideradas en orden descendente por su métrica de similitud y si es aplicable sustituyen al segmento de la transcripción original que corresponda.

Sea T_o la transcripción de audio originalmente producida por el sistema de reconocimiento de voz y $C = \{c_1, \ldots, c_n\}$ un conjunto de n frases específicas del contexto, el Algoritmo 1 modifica T_o para producir una transcripción corregida T_c. Para aplicar este algoritmo es necesario definir las siguientes especificaciones:

- Un criterio para la construcción del conjunto P de palabras fuera de contexto.
- Un tamaño de ventana v de la región vecina de p_j.
- Un mecanismo de construcción del conjunto S_j.
- Una métrica de distancia de edición $d(f_s, f_c)$. En este trabajo se utilizó la distancia Levenshtein.
- El umbral de decisión para la distancia de edición u.

Algoritmo 1 Algoritmo de corrección de transcripciones

Input: La transcripción de audio original T_o, n frases específicas del contexto $C = \{c_1, \ldots, c_n\}$, un umbral de distancia de edición normalizada u, y un tamaño de ventana v de palabras vecinas.

Output: Una transcripción corregida T_c.

1: Inicializar la transcripción corregida $T_c = T_o$.
2: **for all** $c \in C$ **do**
3: Generar la representación fonética f_c de la frase c.
4: **end for**
5: Construir el conjunto $P = \{p_1, \ldots, p_m\}$ con las palabras contenidas en T_o que son consideradas fuera de contexto.
6: **for all** $p_j \in P$ **do**
7: Construir un conjunto S_j de subfrases susceptibles a reemplazo usando v.
8: **for all** $s \in S_j$ **do**
9: Generar la representación fonética f_s de la frase s.
10: Calcular la distancia normalizada de edición $d(f_s, f_c)$ para toda f_c.
11: **end for**
12: Seleccionar el par (s^*, c^*) tal que $\arg\min_{s,c} d(f_s, f_c)$.
13: **if** $d(f_s, f_c) < u$ **then**
14: Agregar (s^*, c^*) al conjunto de candidatos a reemplazo R.
15: **end if**
16: **end for**
17: **if** $R \neq \emptyset$ **then**
18: Ordenar $R = \{(s_1^*, c_1^*), \ldots, (s_L^*, c_L^*)\}$ ascendentemente en $d(f_s, f_c)$.
19: **for** $i = 1$ hasta $i = L$ **do**
20: **if** s_i^* no contiene palabras marcadas en T_o **then**
21: Sustituir s_i^* con c_i^* en T_c.
22: Marcar las palabras componentes de s_i^* en T_o.
23: **end if**
24: **end for**
25: **end if**
26: **return** T_c

La complejidad del Algoritmo 1 es $O(tn)$ y puede calcularse de la siguiente manera. La generación de la representación fonética en la línea 3 corre en tiempo lineal en la longitud de la frase c, por lo que podemos considerarla como una constante f. Esta línea a su vez se ejecuta n veces, dado que se consideran que existen n frases en C. Por lo tanto esta rutina se realiza fn veces.

La construcción del conjunto P en la línea 5 puede realizarse de diferentes formas. Si la solución se implementa comparando todas las combinaciones de los elementos del conjunto P con los elementos de la transcripción T_o, esta construcción requiere entonces mt operaciones.

La construcción del conjunto S_j en la línea 7, para una ventana $v = 1$, requiere de la creación de 4 subfrases de la siguiente manera. Sea la palabra pivote p_j el conjunto S_j se conformaría con las subfrases $\{p_j,\ p_{j-1}p_j,\ p_jp_{j+1},\ p_{j-1}p_jp_{j+1}\}$.

Esta construcción se realiza por cada una de las m palabras de P. Por lo tanto se requiere un total de $4m$ operaciones.

La generación de la representación fonética de la línea 9 requiere el mismo número de ejecuciones que la línea 3, esto es f repeticiones. Dado que esta generación se repite por cada m palabra de P y cada 4 elementos de S_j, el número total de veces que se ejecuta esta operación es $4fm$.

Calcular la distancia de edición en la línea 10 se realiza para cada combinación de los elementos de C con los elementos de S_j, esto es $n \times 4$. A su vez esta rutina se repite m veces ya que se encuentra en el ciclo que comienza en la línea 6. Es decir, este cálculo requiere $4mn$ operaciones.

Seleccionar el par (s^*, c^*) de la línea 12 sólo requiere de igual forma que la línea 10, $n \times 4$ comparaciones, las cuales se ejecutan por cada una de las m palabras en P, ya que está dentro del ciclo que comienza en la línea 6. Es decir, esta operación se realiza $4mn$ veces.

La operación de agregar (s^*, c^*) al conjunto R se realiza en el peor de los casos m veces.

Ordenar ascendentemente los pares (s_i^*, c_i^*) en la línea 18, puede hacerse en $L = m$ pasos a lo mucho, es decir m veces.

Finalmente, cuando $L = m$, la sustitución en la línea 21 y la marcación de palabras en la línea 22 se realizan a lo mucho m veces cada una, es decir en $m + m$ operaciones.

En total se requieren ejecutar $fn+mt+4m+4fm+4mn+4mn+m+m+m+m$ operaciones. Donde f es considerada una constante. Es decir, que la complejidad en el peor de los casos está determinada por el cálculo de la distancia de edición en la línea 10 y la selección del par (s^*, c^*) en la línea 12, las cuales son $4mn$, es decir $O(mn)$, donde m es el número de palabras por ser reemplazadas, siendo el peor de los casos cuando todas las palabras t de la transcripción T_o son reemplazadas, es decir, cuando $m = t$. Por lo tanto llegamos a la conclusión de que la complejidad del Algoritmo 1 es $O(tn)$.

Ahora bien, en un diálogo típico las transcripciones T_o rara vez superan las 50 palabras. Además, los casos en donde todas las palabras de la transcripción requieren ser reemplazadas, contienen normalmente menos de 5 palabras. Esto significa que la corrección de las transcripciones puede realizarse sin ningún problema en tiempo real.

4. Trabajo experimental

Para probar la capacidad de corrección del algoritmo propuesto se implementó una aplicación sobre la plataforma de comunicación *asterisk* que recibe llamadas telefónicas y captura la señal acústica de las frases pronunciadas por el usuario. Se hicieron pruebas con usuarios enunciando frases específicas, cuyas grabaciones fueron posteriormente enviadas al servicio de Google (Google Cloud Speech API) para su reconocimiento.

Se realizaron experimentos utilizando 451 archivos de audio grabados por 9 diferentes usuarios vía telefónica dentro del contexto de levantamiento de pedi-

dos de una pizzería. Los ejemplos fueron simulados tomando como base conversaciones del levantamiento de órdenes del menú de una pizzería real. Como es común, el menú de la pizzería cuenta con diferentes ingredientes, especialidades y paquetes, muchos de los cuales contienen palabras provenientes de idiomas diferentes al español o nombres inventados con características que los hacen difíciles de identificar para sistemas de reconocimiento del habla con modelos de lenguaje de uso general.

Cada uno de los ejemplos fue grabado en un archivo de formato *flac* [13], y se registró en una base de datos la frase real pronunciada por el usuario y las transcripciones obtenidas como resultado de enviar la señal de audio al servicio proporcionado por Google en dos modalidades diferentes; la primera en el modo básico y la segunda incluyendo como contexto 34 frases propias del dominio de la pizzería, que según la documentación de la API de Google se ven favorecidas durante el proceso de reconocimiento, mejorando así la exactitud de los resultados.

Las peticiones a la API de reconocimiento de voz fueron realizadas enviando una petición http por el método POST a la URL https://speech.googleapis.com/v1/speech:recognize con la siguiente configuración en formato *JSON*:

```
{
    config: {
     encoding: 'FLAC',
     sampleRateHertz: 8000,
     languageCode: 'es-US',
     profanityFilter: false,
     maxAlternatives: 1
    },
    audio: {
     content: base64String
    }
}
```

Para el caso de la modalidad de reconocimiento con contexto, a la petición le fue agregada la propiedad *SpeechContext* cuyo valor es un arreglo conteniendo las frases específicas del contexto. Se utilizó un conjunto de 34 frases propias del dominio de la pizzería, de entre una y tres palabras de longitud. Las frases pueden contener variaciones con errores ortográficos pero parecidas fonéticamente a la pronunciación correcta. El objeto *JSON* utilizado tiene la siguiente estructura:

```
{
    config: {
        encoding: 'FLAC',
        sampleRateHertz: 8000,
        languageCode: 'es-US',
        profanityFilter: false,
        maxAlternatives: 1
```

```
    },
    audio: {
        content: base64String
    },
    SpeechContext: [
        barbiquiu,
        buchelati,
        bustarela,
        dipdish,
        extra pepperoni,
        jueves mozzareloso,
        pizza de corazón,
        pizza ragazza,
        ...
    ]
}
```

Las transcripciones obtenidas del servicio de reconocimiento de voz fueron procesadas mediante el algoritmo de corrección fonética descrito en la sección 3. Como contexto C fue usado el mismo conjunto de 34 frases proporcionado al servicio de Google en su modalidad contextual.

Las especificaciones del algoritmo fueron las siguientes:

- El conjunto P corresponde a todas las palabras presentes en la transcripción de entrada que no se encuentran en C y con un tamaño mínimo de 4 caracteres. $P = \{p \mid p \in T_o,\ p \notin C,\ len(p) \geq 4\}$
- El tamaño de ventana fue $v = 1$.
- Para cada palabra p_j se consideraron sus vecinas inmediatas, construyéndose el conjunto de la siguiente manera $S_j = \{p_j,\ p_{(j-1)}p_j,\ p_jp_{(j+1)},\ p_{(j-1)}p_jp_{(j+1)}\}$.
- La función $d(f_s, f_c)$ utilizada fue la distancia de Levenshtein estándar [9] con costo unitario para supresiones, inserciones y sustituciones. Dicha métrica fue normalizada con relación al máximo tamaño de las frases de entrada.
- Se experimentó con diferentes umbrales de decisión u para la función $d(f_s, f_c)$ para observar el impacto de este parámetro en los resultados del algoritmo.

Existen diversos métodos que han sido propuestos para comparar el desempeño de los sistemas reconocedores de voz [7][12], siendo el más común el uso de la métrica WER (*word error rate*). En este trabajo se evaluaron los resultados del algoritmo propuesto usando dicha métrica, entre la frase objetivo y la transcripción hipotética reconocida. Se define WER de la siguiente manera:

$$WER = \frac{S + D + I}{N}, \tag{1}$$

donde S es el número de sustituciones, D el número de supresiones, I el número de inserciones y N el número de palabras en la frase objetivo. Los valores S, D, I y N fueron acumulados globalmente como resultado de calcular el número

de ediciones necesarias para transformar la transcripción hipotética en la frase objetivo correcta para cada uno de los 451 ejemplos.

Al calcular WER para los resultados de ambas versiones del servicio de Google se obtuvieron dos líneas de base que sirven de referencia para evaluar los resultados del algoritmo propuesto. El WER obtenido con la transcripción del servicio básico fue del 33.7 %, mientras que la trascripción con el servicio contextualizado tuvo un WER de 31.1 %. Si bien, estos valores parecen altos con relación a los resultados reportados en los sistemas de reconocimiento de uso general, hay que considerar que el conjunto de frases de ejemplo es muy específico del dominio, lo que aumenta la dificultad del problema. Además, la degradación en la señal producida por el uso de lineas telefónicas convencionales también afecta el desempeño.

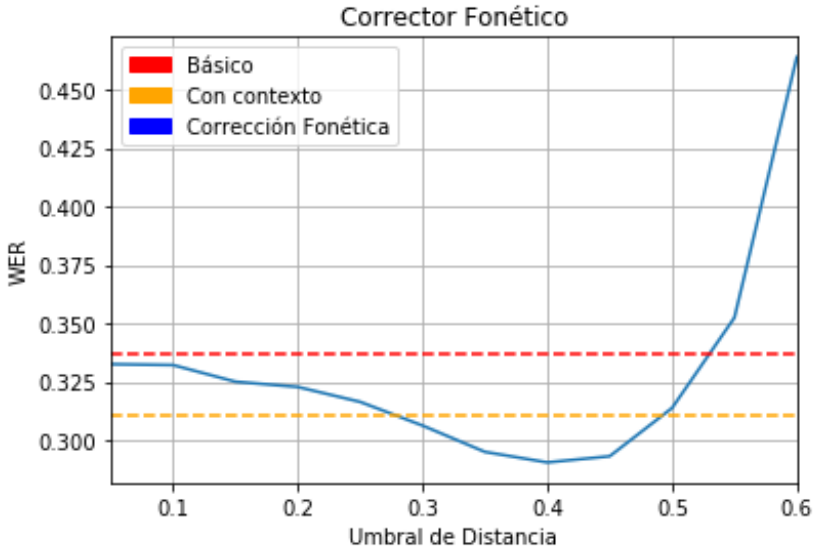

Fig. 1. Resultados del algoritmo para diferentes valores de u tomando como entrada la transcripción básica del servicio TTS de Google.

La Figura 1 muestra el comportamiento del algoritmo al variar el parámetro u cuando se ejecuta tomando como entrada las transcripciones obtenidas con el servicio básico. El mejor resultado se obtiene cuando $u = 0.4$, donde el WER obtenido se reduce al 29.1 %. Tomando como línea de base la transcripción básica, el resultado mejora en un 4.6 % el WER global y el número de sentencias de ejemplo que fueron mejoradas en su reconocimiento fue de 97 de un total 325 frases que fueron reconocidas erróneamente por el servicio de Google.

La Figura 2 presenta los resultados cuando las frases de contexto son enviadas al servicio de Google y las transcripciones procesadas posteriormente por el

algoritmo de corrección fonética. En este caso se observa una mejora en el WER alcanzando un mínimo del 27.3 % también con el parámetro $u = 0.4$. Con este valor el WER mejora en un 3.8 % con relación a la transcripción contextual y un 6.4 % con respecto a la transcripción básica. El número de sentencias mejoradas en este caso fue de 87 de un total de 319 reconocidas erróneamente por Google.

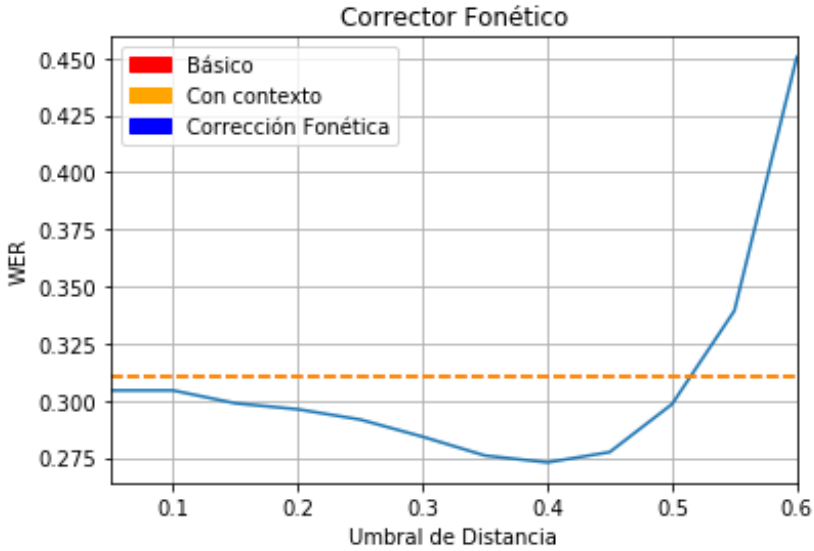

Fig. 2. Resultados del algoritmo para diferentes valores de u tomando como entrada la transcripción con contexto del servicio TTS de Google.

La Tabla 1 señala el número de errores en el reconocimiento para el total de 2664 palabras contenidas en los ejemplos de audio. A partir del reconocimiento básico el algoritmo de corrección fonética produce una reducción en el WER relativo del 12.7 %. Cuando se aplica el algoritmo a la versión con contexto del reconocedor la mejora relativa en WER es del 10.3 %.

Tabla 1. Número de errores de edición y WER relativo.

Modo de reconocimiento	Errores TTS	Errores corrector	WER relativo
Básico	897	774	13.6 %
Con contexto	828	727	12.2 %

Las Figuras 3 y 4 muestran el comportamiento en el porcentaje de sentencias erróneas corregidas al variar el umbral de distancia a partir de las transcripciones en las dos modalidades de reconocimiento de voz. Se observa que en ambos casos

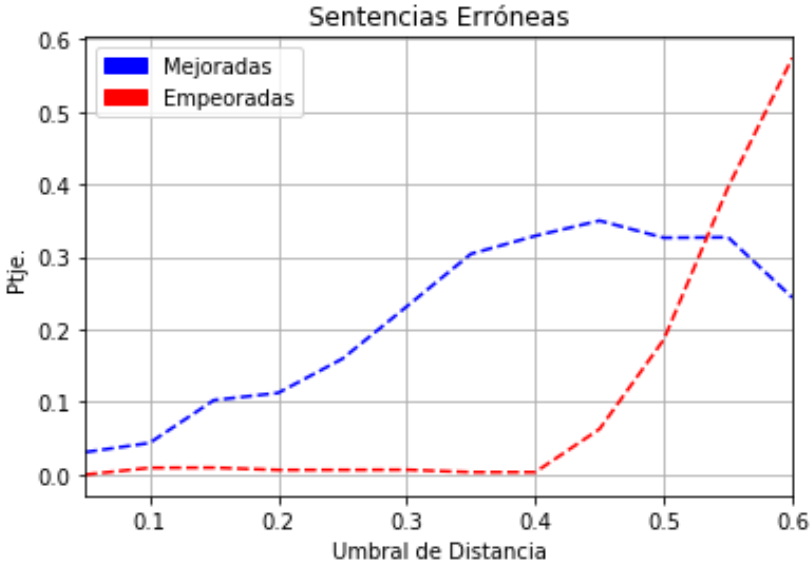

Fig. 3. Porcentaje de sentencias mejoradas con relación al total de sentencias con errores de reconocimiento en la transcripción básica.

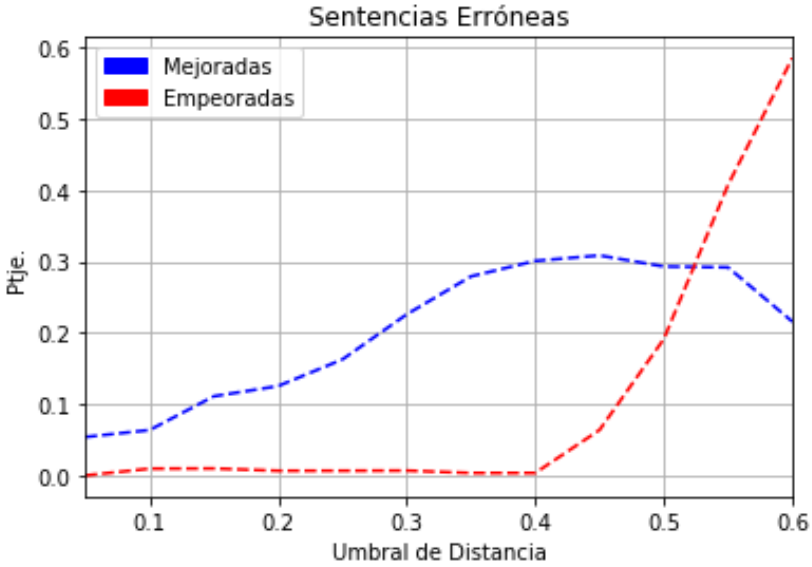

Fig. 4. Porcentaje de sentencias mejoradas con relación al total de sentencias con errores de reconocimiento en la transcripción con contexto.

al superar el valor de $u = 0.4$ las correcciones fonéticas comienzan a empeorar el reconocimiento, en lugar de mejorarlo.

Algunos ejemplos donde el proceso de corrección mejora el reconocimiento se observan en la Tabla 2. Se muestran casos donde el algoritmo logra corregir por completo la frase reconocida por Google, mientras que en otros casos donde la transcripción resultó bastante mala, se logra mejorar lo suficiente como para identificar el contexto de la frase.

Tabla 2. Ejemplos de frases mejoradas por el proceso de corrección fonética. Para cada ejemplo se proporciona la Frase Google (F. G.), la Frase Corregida (F. C.) fonéticamente y finalmente la Frase Objetivo (F. O.).

F. G.	Mándame una Buscar ella
F. C.	Mándame una bustarella
F. O.	Mándame una bustarella
F. G.	Voy a querer una grande de chile ta
F. C.	Voy a querer una grande de chuleta
F. O.	Voy a querer una grande de chuleta
F. G.	2 pizzas medianas y clover
F. C.	2 pizzas medianas meat lover
F. O.	2 pizzas medianas meat lover
F. G.	La pizarra García mediana
F. C.	La pizza ragazza mediana
F. O.	Una pizza ragazza mediana
F. G.	Pistas de Barbie dress up
F. C.	Pizzas de barbecue dress up
F. O.	Dos pizzas de barbecue con mucho queso
F. G.	Quiero un vitel aquí
F. C.	Quiero un Buccellati
F. O.	Quiero un Buccellati
F. G.	Un paquete de jugadores mozzareloso
F. C.	Un paquete de jueves mozzareloso
F. O.	Un paquete de jueves mozzareloso

5. Conclusiones y trabajo futuro

En este artículo se ha propuesto un algoritmo para corregir los errores del reconocedor de voz de Google en aplicaciones de dominios específicos. En los dominios específicos se tiene la ventaja de que es posible generar un diccionario reducido de palabras *ad hoc* a la aplicación y dicho diccionario puede utilizarse para corregir los errores de reconocimiento. En el algoritmo propuesto se utiliza la distancia de Levenshtein sobre fonemas para asignar índices a las palabras candidatas a ser usadas en las correcciones. El algoritmo se probó experimentalmente

para el idioma español, pero es suficientemente general como para emplearse con otros idiomas.

Como caso de estudio se utilizaron frases de un sistema automatizado que toma órdenes de comida para llevar, en restaurantes de pizza. En los resultados experimentales se muestra que con este algoritmo los errores del reconocedor pueden ser reducidos hasta en un 5.8 % con relación a la transcripción básica del servicio de Google. El algoritmo logra mejorar el reconocimiento en alrededor del 30 % de las frases que contienen errores. Aunque no en todos los casos la corrección es perfecta, se logra mejorar lo suficiente como para entender el contexto de la frase, lo cual cobra mayor importancia considerando que dicha transcripción corregida frecuentemente es utilizada como entrada para algoritmos de clasificación de intenciones y reconocimiento de entidades propios de un sistema de entendimiento automatizado de lenguaje natural.

Durante la experimentación se dieron algunos casos donde el algoritmo produce un artefacto erróneo, por ejemplo para la transcripción de entrada "En que consiste el jueves mozart el oso" se obtiene como resultado "En que consiste el jueves mozzareloso oso". Es posible que este tipo de casos puedan ser corregidos mediante diferentes criterios de selección de los conjuntos S_j en el algoritmo, lo que requiere sin duda alguna un análisis más cuidadoso. También se prevé como trabajo futuro explorar diferentes variedades de distancia de edición, incluyendo los costos para diferentes tipos de edición, como las supresiones, inserciones, sustituciones, y transposiciones.

Referencias

1. Anantaram, C., Kopparapu, S.K., Kini, N., Patel, Ch.: Improving ASR Recognized Speech Output for Effective Natural Language Processing. In: ICDS 2015, The Ninth International Conference on Digital Society, pp. 17–21 (2015)
2. Bassil, Y., Alwani, M.: Post-editing error correction algorithm for speech recognition using Bing spelling suggestion. International Journal of Advanced Computer Science and Applications (2012)
3. Becerra, A., de la Rosa, J.I., González, E.: A case study of speech recognition in Spanish: from conventional to deep approach. IEEE ANDESCON (2016)
4. Errattahi, R., El Hannani, A., Ouahmane, H.: Automatic Speech Recognition Errors Detection and Correction: A Review. In: Abbas, M., Abdelali, A. (eds.) 1st International Conference on Natural Language and Speech Processing, Procedia Computer Science, vol. 128, pp. 32–37 (2018)
5. Favre, B., Cheung, K., Kazemian, S., Lee, A., Liu, Y., Munteanu, C., Nenkova, A., Ochei, D., Penn, G., Tratz, S., Voss. C., Zeller, F.: Automatic Human Utility Evaluation of ASR Systems: Does WER Really Predict Performance?. In: Interspeech 2013, 14th Annual Conference of the International Speech Communication Association, pp. 3463–3467 (2013)
6. Fusayasu, Y., Tanaka, K., Takiguchi, T., Ariki, Y.: Word-error correction of continuous speech recognition based on normalized relevance distance. In: International Joint Conference on Artificial Intelligence (2015)
7. Gillick, L., Cox, S.J.: Some statistical issues in the comparison of speech recognition algorithms. In: International Conference on Acoustics, Speech, and Signal Processing (1989)

8. Harwath, D., Gruenstein, A., McGraw, I.: Choosing Useful Word Alternates for Automatic Speech Recognition Correction Interfaces. In: Interspeech 2014, 15th Annual Conference of the International Speech Communication Association, pp. 949–953 (2014)
9. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, Vol. 10, 707 (1966)
10. Li, B., Chang, F., Guo, J., Liu, G.: Speech recognition error correction by using combinational measures. In: IEEE International Conference on Network Infrastructure and Digital Content (2012)
11. Nakatani, R., Takiguchi, T., Ariki, Y.: Two-step Correction of Speech Recognition Errors Based on N-gram and Long Contextual Information. In: Interspeech 2013, 14th Annual Conference of the International Speech Communication Association, pp. 3747–3750 (2013)
12. Pallett, D., Fisher, W., Fiscus, J.: Tools for the analysis of benchmark speech recognition tests. In: International Conference on Acoustics, Speech, and Signal Processing (1990)
13. Sitio web del proyecto Flac. https://xiph.org/flac/, accedido por última vez el 15/05/2018.