UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

TESIS DE GRADO

"DISEÑO DE ALGORITMO DE BUSQUEDA APROXIMADA EN LENGUAS INDIGENAS ORIGINARIAS CAMPESINAS"

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: CIENCIAS DE LA COMPUTACIÓN

POSTULANTE: Univ. José Carlos Laura Ramírez
TUTOR METODOLÓGICO: M. Sc. Edgar Palmiro Clavijo Cardenas
ASESOR: M. Sc. Jorge Humberto Terán Pomier

La Paz – Bolivia

2016

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

AGRADECIMIENTOS

A la Universidad Mayor de “San Andrés”.

A la Carrera de Informática.

Al capitulo estudiantil de la ACM – ICPC.

Un agradecimiento especial a los docentes de la carrera que me ayudaron con este trabajo:

M. Sc. Jorge Terán por darme la idea del trabajo y ayudarme a llevarla a cabo.

M.Sc. Edgar Clavijo por ayudarme con mi trabajo y darme las guías correspondientes.

M.Sc. Lucio Torrico por corregirme y darme consejos, aunque no los haya tomado muy en cuenta.

DEDICATORIA

A Dios por permitirme llegar hasta este momento en mi vida.

A mi familia por apoyarme en todo momento.

A mis “ameguetos” con quienes no solo he compartido la “U” sino una parte importante de mi vida.

A esa persona única y especial que me ha apoyado en esta última parte de mi carrera.

RESUMEN

La búsqueda aproximada en diccionarios consiste en buscar entre las palabras aquellas que se asemejen a un patrón de búsqueda. Si bien la búsqueda aproximada en diccionarios puede ser resuelta por diversos métodos en este trabajo se utiliza el enfoque de la programación dinámica. Se utiliza un algoritmo de programación dinámica para hallar la distancia de Levenshtein o de edición entre palabras y en base a la distancia de edición se forma un conjunto de resultados para la búsqueda.

La búsqueda aproximada con programación dinámica es un problema cuya solución es bien conocida para lenguas con alfabetos estándar (ingles o español), sin embargo cuando se utilizan en lenguas con alfabetos no estándar los algoritmos convencionales no funcionan de la misma manera. En Bolivia existen lenguas indígenas originarias campesinas con alfabetos no estándar y por ello para poder realizar búsqueda aproximada en diccionarios de estas lenguas se presente en este trabajo el algoritmo ABCR2. El algoritmo ABCR2 esta adaptado para el aymara aunque se espera poder extender para otras lenguas.

PALABRAS CLAVES: algoritmo, búsqueda aproximada, distancia de edición, programación dinámica, patrón de búsqueda, algoritmo ABCR2.

ÍNDICE

CAPÍTULO I

1. MARCO REFERENCIAL

1.1. INTRODUCCIÓN

Actualmente se ha visto que no existe buena comunicación digital en lenguas indígenas originarias campesinas del Estado Plurinacional de Bolivia, lo que también provoca dificultad en la generación de material científico, histórico, periodístico, etc., en dichas lenguas, causando con el tiempo que estas se extingan con facilidad (“Pre Cumbre reflexiona sobre problemas que atraviesa la comunicación indígena en el país”, s. f.).

Se ha observado que en el Estado Plurinacional de Bolivia no existe un diccionario oficial de lenguas indígenas originarias campesinas, lo que provoca que no exista un alfabeto estándar para su escritura y los hablantes de dichas lenguas no puedan generar material escrito de una manera estandarizada.

Debido a la falta de estandarización, no existe un método preciso de búsqueda de palabras en diccionarios de lenguas indígenas originarias campesinas, dado que el desarrollo informático no contempla nuestras lenguas originarias.

Por tanto el presente trabajo propone el diseño e implementación del algoritmo ABCR2 para la búsqueda de palabras en diccionarios de lenguas indígenas originarias campesinas, que ayude a las personas a buscar información digital de manera precisa.

De esta manera se apoyará a la producción intelectual en dichas lenguas para que el Estado Plurinacional de Bolivia pueda preservar sus lenguas indígenas originarias campesinas y por ende su cultura.

1.2. ANTECEDENTES

En la última década, se ha observado un incremento del interés en las lenguas indígenas del Estado Plurinacional de Bolivia, sin embargo, un número considerable de las lenguas indígenas que aún son habladas en Bolivia está en serio peligro de extinción y puede ser que se pierdan. ("Number of Endangered Languages by Country"*, 2011*)

En el 2009 se proclama la Constitución Política del Estado Plurinacional de Bolivia, en su artículo 5, inciso primero se reconocen 36 lenguas indígenas como idiomas oficiales, inciso segundo se norma el uso de al menos dos idiomas oficiales en la administración de los gobiernos departamentales y autónomos.

En la Carrera de Informática, de la Universidad Mayor de San Andrés, se desarrolló un trabajo similar enfocado a la corrección ortográfica de la lengua indígena originaria campesina aymara. Corrector Ortográfico del idioma aymara con autómatas finitos, cuyo objetivo es la corrección de textos a nivel ortográfico en el idioma aymara (Saca, 2011).

1.3. PLANTEAMIENTO DEL PROBLEMA

Actualmente no existen medios digitales para la preservación de lenguas indígena originarios campesinas, causando que exista dificultad en la generación de material científico, histórico, periodístico, etc., esto puede generar a largo plazo una extinción de estas lenguas, ya que tampoco existe financiamiento para su preservación. (Apaza, 2012)

Se ha detectado que no existen estándares relacionados a los alfabetos de las lenguas indígenas originarias campesinas. Tampoco existen algoritmos de búsqueda diseñados para estas lenguas y los algoritmos existentes no producen los resultados esperados en el proceso de búsqueda.

Tomemos como ejemplo el aymara. El aymara no tiene una buena correspondencia entre su pronunciación y su escritura, es decir, pese a saber como pronunciar una palabra no

necesariamente se puede saber como escribirla. Por ejemplo una palabra muy común en aymara es “qullqi” que debido a las reglas de pronunciación del aymara es pronunciado como se pronunciaría en español “colque”, el problema es que alguien que no sabe como escribir la palabra podría poner en el buscador “colque”, “kolque”, “qolqe”, etc., sin obtener respuesta alguna.

El problema central de este estudio está dado porque no existe un algoritmo de búsqueda de palabras en diccionarios para las lenguas indígenas originarias campesinas, tomando en cuenta los errores propios de estas lenguas y sus hablantes.

1.4. DEFINICIÓN DE OBJETIVOS

1.4.1. OBJETIVO GENERAL

Diseñar e implementar el algoritmo ABCR2 para la búsqueda de palabras en diccionarios de lenguas indígenas originarias campesinas del Estado Plurinacional de Bolivia.

1.4.2. OBJETIVOS ESPECÍFICOS

- Estudiar la codificación de caracteres en los alfabetos de las lenguas indígenas originarias campesinas.
- Estudiar los errores en la escritura de las lenguas indígenas originarias campesinas.
- Generar las especificaciones para el algoritmo ABCR2.
- Determinar criterios de comparación entre cadenas para alfabetos de lenguas indígenas originarias campesinas.
- Comparar la precisión de los algoritmos de búsqueda aproximada convencionales y el algoritmo ABCR2.

1.5. HIPÓTESIS

El diseño del algoritmo ABCR2 para la búsqueda aproximada en diccionarios de lenguas indígenas originarias campesinas del Estado Plurinacional de Bolivia mejora la precisión de los resultados de algoritmos convencionales en dichas lenguas.

1.5.1. OPERACIONALIZACIÓN DE VARIABLES

Variables independientes

- Diseño del algoritmo ABCR2.
- Búsqueda aproximada en diccionarios de lenguajes indígenas originarias campesinas.

Variables dependientes

- Mejora la precisión de los resultados de algoritmos convencionales en dichas lenguas.

1.6. JUSTIFICACIÓN

1.6.1. ECONÓMICA

Actualmente existen diversas organizaciones dedicadas a preservar los lenguajes que están en peligro de desaparecer.

La UNESCO en su programa “Lenguas en Peligro” (Endangered Languages en inglés) busca apoyar a las comunidades, expertos y gobiernos, dando herramientas para monitoreo, defensa y evaluación sobre las diversas lenguas. (“Projects | United Nations Educational, Scientific and Cultural Organization”, s. f.)

1.6.2. SOCIAL

En el ámbito social el presente trabajo ayudara a que exista una buena comunicación entre lenguas indígenas originarias campesinas y se preserven estas lenguas del Estado Plurinacional de Bolivia.

Bolivia es considerado uno de los lugares más ricos en el mundo en términos de diversidad lingüística. Las 36 lenguas indígenas de Bolivia representan un tesoro lingüístico que merece ser utilizado y promovido en los canales actuales de comunicación digital.

Alrededor de la mitad de las 7.000 lenguas que existen aproximadamente en el mundo está en riesgo de desaparecer durante los próximos 100 años. Pero tenemos a nuestro alcance unas herramientas y una tecnología que podrían evitarlo. Por ello, cualquier aporte que permita la preservación de los lenguajes en peligro, causarán un impacto en el futuro. (“Endangered Languages Project”, s. f.)

1.6.3. CIENTÍFICA

El problema de la búsqueda aproximada en diccionarios para lenguajes con alfabetos bien definidos se encuentra resuelto por diversos métodos, en el presente trabajo se propone el algoritmo ABCR2 para resolver el problema de la búsqueda aproximada en lenguajes sin un estándar en la escritura por falta de un alfabeto propio.

Mediante la definición un modelo de error por falta de un alfabeto estándar se definen criterios para la comparación de cadenas y determinar cuan similares son dos cadenas. Con este nuevo modelo de error se pretende mejorar la precisión de los resultados obtenidos en la búsqueda.

1.7. ALCANCES Y LÍMITES

- Se realizará el diseño y la implementación del algoritmo ABCR2 para la búsqueda de palabras en aymara, una lengua representativa de las lenguas indígenas originarias campesinas del Estado Plurinacional de Bolivia.
- Se definirán criterios para la comparación de cadenas para ajustar los algoritmos convencionales a las necesidades propias de las lenguas indígenas originarias campesinas.
- Se utilizarán diccionarios propios del aymara como entrada para realizar las pruebas al algoritmo.
- La búsqueda se realizará por palabras y no así por frases.
- El idioma aymara es un idioma sufijante (para dar significado agrega sufijos a las palabras). No se tomarán en cuenta los sufijos en la búsqueda, se buscará la palabra en su totalidad.
- El idioma aymara presenta dos fenómenos en su escritura, la elisión y la contracción vocálica. No se toman en cuenta pues ocurren en la unión de dos palabras o en la sufijación.
- No se realizarán pruebas para medir el tiempo de ejecución del algoritmo ABCR2.

1.8. APORTES

1.8.1. PRÁCTICO

El presente trabajo pretende ayudar a toda la población boliviana que desea realizar la búsqueda de información digital en los lenguas indígenas originarias bolivianas, sean estas personas hablantes o no, ya que contarán con un medio para búsqueda de palabras.

1.8.2. TEÓRICO

El presente trabajo implementa los conceptos de búsqueda aproximada en diccionarios, para lo cual se estudiarán las particularidades de los alfabetos de las lenguas indígenas originarias campesinas.

También se estudiarán los problemas que ocurren en el momento de la transcripción de lenguas que no tienen estándares para la escritura y se intentará dar solución a estos mediante la extensión de los métodos de comparación de cadenas.

1.9. METODOLOGÍA

La metodología de investigación que se empleará en el presente trabajo sera el método lógico inductivo, pues vamos a partir de casos particulares de problemas presentados en algún lenguaje indígena y después generalizaremos la solución a los demás lenguajes en cuestión.

CAPÍTULO II

2. CONCEPTOS LINGÜÍSTICOS Y OPERACIONES EN CADENAS PARA LA BÚSQUEDA APROXIMADA

En este capítulo presentaremos primeramente los conceptos lingüísticos necesarios para entender las lenguas indígenas originarias campesinas del Estado Plurinacional de Bolivia y posteriormente presentaremos los conceptos básicos necesarios para poder explicar el problema de la búsqueda aproximada, se definirá la notación a seguir durante el desarrollo del presente trabajo así como los métodos que existen para resolver el problema de la búsqueda aproximada.

2.1. CONCEPTOS LINGÜÍSTICOS

2.1.1. LENGUAJE

El lenguaje es el medio de comunicación entre seres humanos, los cuales utilizan signos orales y escritos, sonidos y gestos que poseen un significado. El lenguaje también puede entenderse como la capacidad humana que permite conformar el pensamiento.

Otra definición de lenguaje es la habilidad de asimilar y usar sistemas complejos de comunicación que permite a los humanos el intercambio verbal o simbólico de expresiones con significado.

2.1.2. LENGUA

Tomaremos como referencia a Ferdinand de Saussure (1913) para entender que es una lengua. Saussure define un signo lingüístico como la relación que existe entre una imagen acústica, que es la forma en que el hablante va a pronunciar un nombre, y un concepto mental. Entonces según Ferdinand, una lengua es un sistema de signos lingüísticos, voz, escritura o gestos, que permiten la comunicación entre las personas de una misma comunidad lingüística.

Diremos entonces que una lengua es el conjunto de signos lingüísticos que están a disposición de todos los hablantes de un mismo idioma. Además, la lengua engloba dos funciones sociales principales, por un lado la comunicación y por el otro la identificación.

2.1.3. HABLA

El habla es el uso de la lengua que un hablante hace en un mensaje determinado. Por extensión, llamamos también habla a la manera de usar la lengua de una determinada comunidad.

Si bien el habla es un acto individual este depende de la lengua que es social.

2.1.4. FONOLOGÍA

La fonología es una subcampo de la lingüística encargada de la organización sistemática de los sonidos en los lenguajes. La fonología describe el modo en que los sonidos funcionan, en una lengua en particular o las lenguas en general, en un nivel abstracto o mental.

La fonología estudia los elementos fónicos de una lengua desde el punto de vista de su función. En español, sabemos que hay una unidad /b/ en beso, porque si la cambiamos por /p/ obtenemos otra palabra: peso. En este caso, las unidades /b/ y /p/ se llaman fonemas.

2.1.4.1. GRAFEMA

Los grafemas son las unidades mínimas de la lengua escrita. Son modelos que se construyen en la mente del hablante de la lengua y, al igual que los fonemas distinguen su significado.

2.1.4.2. FONEMA

Un fonema es la abstracción (imagen mental) de los sonidos del habla humana. Las principales características de los fonemas son que:

- Son abstractos, invisibles y carecen de significado.
- Son las unidades básicas del estudio fonológico de una lengua.
- Se representan gráficamente dentro de dos rayas oblicuas: /fonema/.
- Pueden ser correspondidos por varios grafemas.
- Sus sonidos se denominan alófonos y se representan entre corchetes.

2.1.4.2.1. CLASIFICACIÓN DE LOS FONEMAS CONSONÁNTICOS

Según el punto de articulación:

- Bilabiales: los dos labios se tocan - /p/, /b/, /m/
- Labiodental: labio inferior y dientes superiores se tocan - /f/
- Interdental: lengua entre los dientes - /θ/
- Dental: lengua detrás de los dientes superiores - /t/, /d/
- Alveolar: lengua sobre la raíz de los dientes superiores - /s/, /l/, /r/, /rr/, /n/
- Palatal: lengua y paladar - /ch/, /y/, /ñ/
- Velar: lengua y velo del paladar - /k/, /g/, /j/

Según el modo de articulación:

- Oclusivo: cierre total y momentáneo - /p/, /b/, /t/, /d/, /k/, /g/, /n/, /m/
- Fricativo: estrechamiento por donde pasa el aire rozando - /f/, /θ/, /j/, /s/
- Africado: se produce una oclusión y después una fricación - /ch/, /ñ/
- Lateral: el aire pasa rozando los lados de la cavidad bucal - /l/, /y/
- Vibrante: el aire hace vibrar la punta de la lengua al pasar - /r/, /rr/

Según acción de las cuerdas vocales:

- Sordo: no vibran las cuerdas vocales - /p/, /t/, /k/, /ch/, /θ/, /s/, /j/, /f/
- Sonoro: sí vibran las cuerdas - /b/, /d/, /l/, /r/, /rr/, /m/, /n/, /y/, /g/

Según fonemas nasales y orales:

- Nasal: parte del aire pasa por la cavidad nasal - /m/, /n/, /ñ/
- Oral: todo el aire pasa por la boca - El resto

2.1.4.2.2. CLASIFICACIÓN DE LOS FONEMAS VOCÁLICOS

Según el punto de articulación:

- Anteriores: /e/, /i/
- Medio o central: /a/
- Posteriores: /o/, /u/

Según el modo de articulación:

- Abertura máxima o Abierto: /a/
- Abertura media o Semiabiertos: /e/, /o/
- Abertura mínima o Cerrados: /i/, /u/

2.1.5. TRANSCRIPCIÓN

La transcripción en el sentido lingüístico, es la representación de un lenguaje en su forma escrita. Las fuentes pueden ser las expresiones (habladas o lenguajes de señas) o textos pre escritos.

El lenguaje escrito no es más que una idealización, hecho a base de limitado conjunto de símbolos distintos y discretos. El lenguaje hablado, por otra parte, es un fenómeno continuo, hecho de un número potencialmente ilimitado de componentes. No existe un sistema predeterminado para distinguir y clasificar estos componentes, y consecuentemente, no hay una manera de hacer corresponder estos componentes en los símbolos escritos.

2.1.6. LENGUAS INDÍGENAS ORIGINARIAS CAMPESINAS

En Bolivia desde la promulgación de la Constitución Política del Estado, en febrero del 2009, fue instaurada la autonomía indígena originaria campesina. Dicha autonomía reconoce el gobierno propio de las naciones y pueblos indígenas originarios campesinos en el marco de la libertad, dignidad, tierra y respeto a su identidad y formas de organización propia.

En el artículo 30 inciso primero, se especifica: "Es nación y pueblo indígena originario campesino toda la colectividad humana que comparta identidad cultural, idioma, tradición histórica, instituciones, territorialidad y cosmovisión, cuya existencia es anterior a la invasión colonial española". Así pues, una lengua indígena originaria campesina es toda aquella lengua utilizada por una nación y pueblo indígena originario campesino.

2.1.6.1. LENGUA AYMARA

La lengua aymara procede de los Andes centrales, de las serranías del Perú. Se fue extendiendo hacia el sur y fue adoptada como lengua materna por algunos pueblos. Posteriormente fue reemplazada por el quechua desde la costa hasta el Cuzco y alrededores, aunque fue ampliamente hablada desde Arequipa hasta Poopó a la llegada de los conquistadores españoles.

La lengua aymara hoy es hablada por más de dos millones de hablantes ubicados en los países de Bolivia, Perú, Chile y Argentina. El aymara en Bolivia se habla en el nor-oeste y la parte nor-central, sobre todo en la ciudad de La Paz.

En el caso de Bolivia, según el censo (INE 2001), los 1.462.285 hablantes (mayores de 6 años) conformaban el 20.97% de la población nacional, en el último censo (INE 2012) de los 10.027.254 habitantes se denominan aymaras 1.191.352 y conforman 11.18% de la población nacional.

2.1.6.1.1. ALFABETO

Mediante el DECRETO SUPREMO Nº 20227 de mayo de 1984, se declara oficialmente el ALFABETO ÚNICO, para la escritura de los idiomas quechua y aymara.

Tabla 1.

Vocales del aymara

Anterior **i** Cerrada		**u** Posterior Cerrada
Semi abierta **[e]**		**[o]** Semi abierta
	a Central Abierta	

Nota. Extraído de "Vocabulario de la lengua aymara" de Ludovico Bertonio (1612) re-publicación por ILLA-A (2011).

Tabla 2

Consonantes del aymara

MODO DE ARTICULACIÓN	PUNTO DE ARTICULACIÓN					
	Bilabiales	Dentales	Alveolares	Palatales	Velares	Pos-velares
Oclusivas simples	P	T			K	Q
Oclusivas espiradas	PH	TH			KH	QH
Oclusivas glotalizadas	P'	T'			K'	Q'
Africada simple				CH		
Africada espirada				CHH		
Africada glotalizada				CH'		
Fricativa			S		J	X
Laterales			L	LL		
Nasales	M		N	Ñ		
Semiconsonantes	W			Y		
Vibrante simple			R			

Nota. Extraído de "Vocabulario de la lengua aymara" de Ludovico Bertonio (1612) republicación por ILLA-A (2011).

2.1.6.1.2. GRAMÁTICA

El alfabeto fonémico del aymara, esta compuesto por 30 fonemas segmentales: 26 consonantes, 3 vocales y un alargamiento vocálico.

Los fonemas consonánticos a su vez se dividen en sordos y sonoros. Las consonantes sordas son 15 oclusivas y 3 fricativas. Los fonemas sonoros comprenden a la serie de los nasales, laterales, vibrante simple y semiconsonantes. El alargamiento vocálico tiene un estado fonémico y como tal sirve para distinguir fonemas.

2.1.6.1.3. REGLAS DE PRONUNCIACIÓN

Como podemos ver en la tabla del alfabeto único, el aymara no posee las vocales e y o, aunque, en la pronunciación si utilizan estos sonidos. Esto se debe a que en el aymara la diferencia en la pronunciación de la /e/ y la /i/ nunca es importante, por que nunca cambia el sentido de una palabra, lo mismo pasa con la /o/ y la /u/ (Heggarty, 2006). Esto se da cuando las vocales cerradas /i/, /u/ se encuentran antes o después de los fonemas pos velares /q/, /qh/, /q'/ y /x/.

En la pronunciación del aymara existen consonantes que tienen un sonido muy similar, por ejemplo, /q/, /qh/, /q'/, /k/, /kh/ y /k'/ y es difícil de corresponder a un sonido similar en la lengua española, pues para todos ellos solo tenemos el sonido de la /c/.

Un punto importante a tomar en cuenta, es que en el aymara, no existen los diptongos o triptongos, es decir, no puede existir la sucesión entre dos vocales.

2.2. ERRORES EN LA ESCRITURA

La mayoría de los lenguajes con un sistema de escritura alfabético tienen ortografía, es decir, una manera correcta de escribir las palabras.

En los modelos tradicionales de aprendizaje, es asumido que los que aprenden saben como hablar el lenguaje que deben escribir. Si la escritura estándar de cada lenguaje tuviera una correspondencia uno a uno entre los fonemas y los grafemas, la escritura sería una tarea trivial: sabiendo la pronunciación correcta de una palabra y los grafemas de un alfabeto, sería intuitivo y fácil inferir la escritura correcta de una palabra.

Aunque solo un pequeño grupo de lenguajes escritos pueden considerar tener una correspondencia uno a uno entre los fonemas y los grafemas como el swahili y tagalog. Otros lenguajes (e.g. italiano y alemán) tienen buena, pero incompleta correspondencia, muchos de los errores por principiantes son por ello concentrados en las áreas de poca correspondencia. Varios lenguajes, incluido el inglés, tienen una ortografía en gran medida convencional. Teniendo varias formas de escribir un mismo sonido, sabiendo la pronunciación correcta de una palabra no da al hablante el indicio suficiente para saber como escribirla.

2.2.1. ERRORES TIPOGRÁFICOS

Los errores tipográficos son comunes en los textos ingresados por teclado, especialmente si el texto es ingresado por mecanógrafos poco experimentados.

Para el presente trabajo, tomaremos en cuenta dos tipos de errores tipográficos comunes los errores mecánicos de tipografía y los errores lingüísticos.

2.2.1.1. ERRORES MECÁNICOS DE TIPOGRAFÍA

Este tipo de error ocurre al momento de ingresar los textos por teclado y es causado simplemente por que se han presionado las teclas equivocadas o en el orden incorrecto, causando accidentalmente transposiciones, inserciones, borrados o substituciones de caracteres. Este tipo de error, también es llamado errata.

2.2.1.2. ERRORES LINGÜÍSTICOS

Este es un error ortográfico particular causado por la interferencia lingüística. Cuando el que escribe el texto no es hablante nativo del lenguaje que está escribiendo, los estándares y las costumbres de su lengua madre pueden aparecer en el texto. También puede darse el caso que el hablante si está escribiendo su lengua madre, pero no sabe como escribirla, ya sea por que no recibió la educación necesaria o simplemente porque no sabe como escribir los caracteres propios de su lengua.

Por ejemplo, consideremos los siguientes errores comunes en el español, dejando de lado su significado y sabiendo que suenan igual.

- Ay, hay, ahí.
- Haya, allá y halla.
- Valla, vaya
- Vasta, basta.

Como se puede ver en la lista tenemos palabras homófonas, que si bien suenan igual se escriben de manera diferente. Estas palabras son claros ejemplos de los errores lingüísticos.

2.3. CONCEPTOS BÁSICOS DE OPERACIONES CON CADENAS

2.3.1. DEFINICIONES Y SUPUESTOS

Sea $\Sigma = \{\Sigma_i\}$ un alfabeto finito ordenado de tamaño $|\Sigma|$. Una cadena es una secuencia finita de caracteres sobre Σ. El conjunto de todas las cadenas de tamaño n sobre Σ se denota por Σ^n, mientras $\Sigma^* = \sum_{n=1}^{\infty} \Sigma^n$ representa el conjunto de todas las cadenas. La cadena vacía será representada por ε. Para cualquier cadena $s \in \Sigma$, su tamaño es denotado por $|s|$. Una serie de variables de tipo cadena y/o carácter, representan su concatenación.

Denotamos el i-ésimo carácter de la cadena s como s[i]. Una subsecuencia de caracteres contiguos de una cadena es una subcadena. La subcadena de s que empieza en la posición i y termina en la posición j es denotada por s[i:j] = s[i]s[i+1]...s[j]. La cadena invertida de s, s[n]s[n-1]...s[1] es denotada como rev(s).

Asumiendo que la cadena s es representada como la concatenación de tres subcadenas posiblemente vacías, s_1, s_2, s_3 (i.e. $s = s_1s_2s_3$). Entonces, la subcadena s_1 es un prefijo de s, mientras que la subcadena s_3 es un sufijo de s.

Una subcadena de tamaño fijo q es llamado q-grama (también llamado n-grama). Los q-gramas de tamaño uno, dos y tres tienen nombres especiales, unigramas, bigramas y trigramas. Considere una cadena s de tamaño n. Introducimos la función q-grama(s) que toma la cadena s y produce una secuencia de n – q + 1 q-gramas contenidas en s:

$$\text{q-grama(s)} = s[1:q], s[2:q+1], ..., s[n-q+1:n]$$

Si n < q, q-grama(s) produce la secuencia vacía. Puede verse que q-grama(s) es esencialmente un mapeo de Σ^* hacia el conjunto de cadenas sobre el alfabeto Σ^q, que es compuesto de todos los posibles q-gramas.

La cadena puede ser transformada en un vector de frecuencia (también se le llama vector de frecuencia de unigrama). El vector de frecuencia es un vector de tamaño $|\Sigma|$, donde el i-ésimo elemento contiene el numero de ocurrencias del i-ésimo carácter del alfabeto en la cadena s. El vector de frecuencia producido por la cadena s es denotado por vect(s).

Ya que q-grama(s) es en si mismo una cadena sobre el alfabeto Σ^q, puede ser transformado en un vector de frecuencia vect(q-grama(s)). Para distinguirlo del vector de frecuencia obtenido directamente de la cadena s, usamos el término vector de frecuencia del q-grama.

Una signatura es una variante del vector de frecuencia. Es un vector binario de tamaño $|\Sigma|$ donde el i-ésimo elemento es igual a 1, si y solo si, el i-ésimo carácter del alfabeto pertenece a s, y 0 en otro caso. La signatura producida por la cadena s es denotada por signat(s).

Varios métodos estudiados reducen la dimensionalidad del problema de la búsqueda proyectando el alfabeto original Σ a un alfabeto más pequeño σ usando una función de hasheado h(c). El alfabeto σ es llamado alfabeto reducido. La función de hasheado produce en términos de caracteres una proyección del conjunto de cadenas sobre el alfabeto original Σ al conjunto de cadenas sobre el alfabeto reducido σ de una manera directa. Dada una cadena s de tamaño n, una proyección correspondiente h(s) está dada por:

$$h(s[1]s[2]...s[n]) = h(s[1])h(s[2])...h(s[n]) \quad (1)$$

2.3.2. DISTANCIA DE EDICIÓN

La historia de la distancia de edición comienza con Damerau (1964) quien presenta errores de ortografía y un método para corregir un solo error. Damerau se enfocó en la inserción, borrado, substitución y transposición de un solo carácter, que representaba el 80% de los errores ortográficos en los textos tecleados. Independientemente, Levenshtein (1966) propuso una función similar definida como el mínimo número de inserciones, borrados y substituciones (pero no transposiciones) necesarias para obtener una cadena a partir de otra.

2.3.2.1. OPERACIONES BÁSICAS DE EDICIÓN Y GUIÓN DE EDICIÓN

Las transformaciones atómicas de subcadenas, como las mencionadas en la sección anterior inserción, borrado, substitución y transposición, son las operaciones básicas de edición. Un guión de edición es una secuencia de operaciones básicas de edición para transformar una cadena en otra.

Una operación básica de edición que consiste en mapear la cadena u en la cadena v es representada por u → v (por simplicidad se obviará la posición donde las operaciones básicas de edición son llevadas a cabo). Denotamos el conjunto de operaciones básicas de edición por **B**.

El conjunto de operaciones básicas de edición usualmente es restringido a las siguientes operaciones de un carácter.

- Inserción: ε → b.
- Borrado: a → ε.
- Substitución: a → b (reemplazo).

En algunos casos **B** es expandido para incluir transposiciones, que consisten en la inversión de caracteres adyacentes: ab → ba.

PROPIEDAD 1. Asumimos que **B** satisface lo siguiente.

- Si u → v ∈ **B**, entonces la operación inversa también pertenece a **B** (simetría).
- a → a ∈ **B** (la operación identidad entre caracteres pertenece a **B**).
- **B** es completo: para cualquier par de cadenas p y s, siempre existe un guión de edición que transforma p en s.

2.3.2.2. DEFINICIÓN DE DISTANCIA DE EDICIÓN

La similaridad entre dos cadenas puede ser expresada a través del tamaño del guión de edición que hace a las cadenas iguales.

DEFINICIÓN 1. Dado un conjunto de operaciones básicas de edición, la distancia de edición ED(p, s) es igual al tamaño del guión de edición más corto que transforma la cadena p en la cadena s. Un guión de edición más corto que transforma p en s es un guión de edición óptimo. Si el conjunto de operaciones básicas de edición contiene solo inserciones, borrados y substituciones, es la distancia de Levenshtein. Si, además el conjunto de operaciones básicas de edición incluye transposiciones, es la distancia de Damerau-Levenshtein.

La distancia de edición puede ser interpretada como el mínimo costo al cual una cadena puede ser transformada en otra. Puede ser generalizada en dos formas:

Primero, a las operaciones básicas de edición les podemos asignar costos individuales $\delta(a \rightarrow b)$, extendiendo la función de costo a un guión de edición $E=a_1 \rightarrow b_1, a_2 \rightarrow b_2, \ldots, a_{|E|} \rightarrow b_{|E|}$ definiendo $\delta(E) = \sum \delta(a_i \rightarrow b_i)$. Ahora nuestra distancia de la cadena p a la cadena s, es el mínimo costo de todos los guiones de edición que transforman p en s. Esta variante de distancia de edición es comúnmente referida como distancia de Levenshtein generalizada.

Segundo, el conjunto de operaciones básicas de edición **B** puede ser extendido para permitir substituciones de subcadenas arbitrarias con pesos asociados en vez de operaciones de edición de caracteres individuales. Esta variante es denominada distancia de edición extendida.

DEFINICIÓN 2. Dado un conjunto de operaciones básicas de edición **B** y una función δ(), la cual asigna costo a todas las operaciones básicas de edición de **B**, la distancia de edición genérica entre la cadena p y la cadena s es definida como el mínimo costo de un guión de edición que transforma p en s.

PROPIEDAD 2. Asumimos que la función de costo $\delta(u \rightarrow v)$ satisface lo siguiente.

- $\delta(u \rightarrow v) \in IR$ (la función de costo tiene valor real).
- $\delta(u \rightarrow v) = \delta(v \rightarrow u)$ (simetría)
- $\delta(u \rightarrow v) \geq 0$, $\delta(u \rightarrow u) = 0$, y $\delta(u \rightarrow v) = 0 \Rightarrow u = v$ (definitud positiva)
- $\forall \gamma > 0$ el conjunto de operaciones básicas $\{ u \rightarrow v \in \mathbf{B} \mid \delta(u \rightarrow v) < \gamma \}$ es finito.

Nótese que la última propiedad implica que **B** es finito.

2.3.2.3. DISTANCIA DE EDICIÓN RESTRINGIDA

El guión de edición restringido no contiene operaciones de edición superpuestas y no modifica una subcadena dos veces. La distancia de edición restringida es el mínimo costo de un guión de edición restringido que transforma la cadena p en la cadena s.

2.3.2.4. ALINEAMIENTO ÓPTIMO

El problema del cálculo eficiente de la función no restringida de Damerau-Levenshtein fue resuelto por Lowrance y Wagner (1975).

Sean las cadenas p y s particionadas en el mismo número de subcadenas posiblemente vacías: $p = p_1p_2...p_L$ y $s = s_1s_2...s_L$, de tal manera que $p_t \rightarrow s_t \in \mathbf{B}$. Adicionalmente asumimos que p_t y s_t no pueden ser vacías al mismo tiempo. Decimos que esta partición define un alineamiento $A = (p_1p_2...p_L, s_1s_2...s_L)$ entre p y s, donde la subcadena p_t esta alineada con la subcadena s_t.

El alineamiento representa el guión de edición restringido $E = p_1 \rightarrow s_1, p_2 \rightarrow s_2, \ldots, p_L \rightarrow s_L$. Definimos el costo del alineamiento A como el costo del correspondiente guión de edición y lo denotamos como:

$$\delta(A) = \sum_{t=1}^{L} \delta(p_t \rightarrow s_t) \qquad (2)$$

Un alineamiento óptimo es un alineamiento con el mínimo costo.

2.3.2.5. CÁLCULO DE LA DISTANCIA DE EDICIÓN

El algoritmo de programación dinámica para calcular el costo de un alineamiento óptimo fue descubierto independientemente por varios investigadores en varios contextos, incluyendo el reconocimiento de voz (Vintsyuk, 1968; Velichko y Zagoruyko, 1970; Sakoe y Chiba, 1971) y en biología computacional (Needleman y Wunsch, 1970). Dejando de lado su descubrimiento temprano, el algoritmo fue desconocido hasta antes de la publicación de Wagner y Fischer (1974) en una revista de ciencias de la computación.

El principio del algoritmo es expresar el costo del alineamiento entre la cadena p y la cadena s usando costo de alineamiento entre sus prefijos. Considere el prefijo p[1:i] de tamaño i y el prefijo s[1:j] de tamaño j de las cadenas p y s, respectivamente. Asumamos que $A = (p_1p_2...p_L, s_1s_2...s_L)$ es un alineamiento óptimo entre p[1:i] y s[1:j], cuyo costo es denotado como $C_{i,j}$.

Usando la ecuación (2), y la definición de alineamiento óptimo es fácil demostrar que $C_{i,j}$ puede ser calculado usando la siguiente recurrencia genérica (Ukkonen, 1985a; Veronis, 1988):

$$C_{0,0} = 0$$
$$C_{i,j} = \min \{ \delta(p[i':i] \rightarrow s[j':j]) + C_{i'-1,j'-1} \mid p[i':i] \rightarrow s[j':j] \in \mathbf{B} \} \quad (3)$$

- El costo del alineamiento óptimo entre la cadena p y la cadena s es igual a $C_{|p|,|s|}$.
- Todos los alineamientos óptimos pueden ser recuperados volviendo hacia atrás en la recursión.

Consideremos ahora el caso de la distancia de Levenshtein, donde p[i':i] → s[j':j] es un costo unitario por carácter insertado, borrado o substituido. Entonces:

$$\delta(p[i':i] \rightarrow s[j':j]) = [\, p[i':i] \neq s[j':j] \,]$$

donde [X] es igual a 1, si la condición X es verdad y es 0 en otro caso. Más aún, hay tres posibles combinaciones de i', j', que corresponden al borrado, inserción o substitución, respectivamente

- i = i − 1 y j = j. (Borrado)
- i = i y j = j − 1. (Inserción)
- i = i − 1 y j = j − 1. (Substitución)

Considerando estas simplificaciones, podemos escribir la ecuación (3) para la distancia de Levenshtein como sigue:

$$C_{i,j} = \begin{cases} C_{0,0} = 0, & i = 0, j = 0 \\ C_{i,j} = C_{i-1,j} + 1, & i > 0 \\ C_{i,j} = C_{i,j-1} + 1, & j > 0 \\ C_{i,j} = C_{i-1,j-1} + \left[p_{[i]} \neq s_{[j]} \right] & i > 0, j > 0 \end{cases} \quad (4)$$

De la recurrencia (4), se puede ver que la matriz de programación dinámica puede ser calculada de arriba hacia abajo y de izquierda a derecha en tiempo y espacio proporcional a $O(|p|.|s|)$.

2.3.3. DISTANCIA DE FRECUENCIA

Consideremos un conjunto de datos compuesto de vectores m-dimensionales. Dado un vector de consulta z, el problema de la búsqueda aproximada en espacios vectoriales consiste en encontrar elementos similares a z. Dependiendo de la definición de similitud, el problema tiene muchas variaciones, entre las cuales están consultas de igualdad parcial y consultas de región (también conocidos como consultas de rangos ortogonales). El problema de satisfacer consultas de distancia de frecuencia es poco conocido, pero provee la base para varios métodos de búsqueda con filtrado.

La distancia de frecuencia se origina del concepto de filtrado por conteo introducido por Grossi y Lucio (1989) con el propósito de la igualdad de subcadenas. Este concepto fue mejorado por varios investigadores incluyendo Jokinen et al. (1996) y Navarro (1997).

Grossi y Lucio probaron lo siguiente: Si el patrón p iguala una subcadena t[i:j] con, a lo más k errores, entonces la subcadena t[j-|p|+1, j] incluye al menos |p| - k caracteres de p (tomando en cuenta las múltiples ocurrencias).

En el caso de la igualdad de palabras completas, el filtrado por conteo puede ser usado recíprocamente: Si ED(p, s) $\leq$ k , entonces la cadena p debe contener al menos |s| - k caracteres de la cadena s y la cadena s debe contener al menos |p|-k caracteres de la cadena p.

2.3.3.1. DEFINICIÓN DE DISTANCIA DE FRECUENCIA

La distancia de frecuencia positiva $FD_{+}(x,y)$ y la distancia de frecuencia negativa $FD_{-}(x,y)$ entre x e y está definida por la siguiente fórmula:

$$FD_{+}(x,y)=\sum_{x_i>y_i} x_i-y_i=\sum_{i=1}^{m}[x_i>y_i]\cdot(x_i-y_i) \tag{5}$$

$$FD_{-}(x,y)=\sum_{x_i<y_i} y_i-x_i=\sum_{i=1}^{m}[x_i<y_i]\cdot(y_i-x_i) \tag{6}$$

$FD(x,y)=max(FD_{+}(x,y),FD_{-}(x,y))$ es la distancia de frecuencia.

2.3.3.2. RELACIÓN CON LA DISTANCIA DE EDICIÓN

Dadas las cadenas p y s, la distancia de frecuencia entre sus correspondientes vectores de frecuencia de unigrama vect(p) y vect(s) es una cota inferior para ambas la distancia de edición restringida y la no restringida entre p y s (Kahveci y Singh, 2001).

$$FD(vect(p), vect(s)) \leq ED(p, s) \tag{7}$$

La desigualdad (7) indica que el problema de la búsqueda aproximada en diccionarios puede ser reducida a buscar un vector espacial. En este enfoque, una cadena es representada por su vector de frecuencia de unigrama. Ya que la distancia de frecuencia entre los vectores de frecuencia de unigrama es una cota inferior para la distancia de edición entre las respectivas cadenas, podemos realizar consultas de distancia de frecuencia de unigramas en el espacio vectorial con el mismo valor límite k. Dado un patrón p y una distancia de edición máxima admitida k, el procedimiento de búsqueda encuentra todas las cadenas tal que:

$$FD(vect(p), vect(s)) \leq k$$

Nótese que la proyección de las cadenas a los correspondientes vectores de frecuencia no preserva la distancia de edición. Proyectar las cadenas a los vectores de frecuencia es un ejemplo de un método de filtrado que consiste en usar un algoritmo computacionalmente menos costoso para filtrar la mayor parte del conjunto de datos.

CAPÍTULO III

3. DISEÑO E IMPLEMENTACIÓN DEL ALGORITMO ABCR2

En el capítulo anterior expusimos algunos conceptos importantes como base para poder definir el problema de la búsqueda aproximada en lenguas indígenas originarias campesinas.

Primeramente veremos cuales son las diferencias entre una lengua indígena originaria con respecto a, por ejemplo, la lengua española. Establecidas las particularidades de las lenguas indígenas originarias campesinas procederemos a listar los problemas encontrados y desarrollaremos el modelo.

3.1. DEFINICIÓN DEL PROBLEMA

Un gran problema de las lenguas indígenas originarias campesinas es que no tienen un alfabeto para su escritura de acuerdo a sus necesidades, de hecho como habíamos definido la transcripción de una lengua es una mera idealización (véase la sección 2.1.5). Además no existe una cultura de producción escrita en lenguas indígenas originarias campesina.

También se mencionó que la pronunciación de las lenguas pese a tener reglas definidas pueden variar de una región a otra y como habíamos visto en las reglas de pronunciación (véase sección 2.1.6.1.3) un mismo sonido puede ser correspondido por más de un grafema.

El problema a resolver entonces es: dado un conjunto de palabras en un diccionario W y un patrón de búsqueda p, posiblemente con errores tipográficos, se trata de encontrar un subconjunto de palabras de W que igualen a p con, a lo más, k errores de tipo mecánico.

3.2. DESARROLLO DE UN MODELO

El algoritmo de búsqueda aproximada ABCR2 debe poder encontrar un subconjunto de palabras del diccionario W que igualen a un patrón de búsqueda p con, a lo más, k errores de tipo mecánico. Como se había visto en el capítulo anterior la distancia de Levenshtein ayuda a calcular dicho subconjunto, si es es que el conjunto de operaciones básicas de edición **B** está compuesto únicamente por inserciones, borrados y substituciones de un solo carácter. Lo que hace distinto al problema planteado es que se tomará en cuenta los errores de tipo mecánico pero no así los errores de tipo lingüístico.

En la sección 2.1.6.1.3 habíamos visto que: en el aymara y en el quechua a un fonema puede corresponderle más de una consonante. Por ejemplo, la palabra waka del quechua, que designa a las sacralidades fundamentales, puede ser escrita como: huaca, waca, guaca, wak'a, por errores tipográficos lingüísticos. Debemos considerar todas las anteriores palabras como iguales, es decir, su distancia de edición debe ser cero.

3.2.1. REGLAS DE REEMPLAZO DE CARACTERES

Como se ha definido el problema, asumimos que las cadenas del diccionario W están bien escritas, por otro lado no podemos decir lo mismo del patrón de búsqueda, el patrón de búsqueda puede tener errores ortográficos.

Como lo que se intenta es no penalizar una palabra que se encuentre mal escrita por errores lingüísticos, el algoritmo durante la comparación entre caracteres deberá considerar iguales los caracteres cuyas transcripciones sean iguales.

Por ejemplo, consideremos la palabra del aymara “qullqi”, que significa dinero o plata, su pronunciación podría ser escrita como “colque”, incluso esta pronunciación es utilizada como un apellido en la región andina.

Si bien estas dos palabras son escritas de manera muy diferente debemos considerarlas como iguales. Al desglosar la palabra qullqi, podemos ver que:

- La [q] suena como una [c].
- De las reglas de la pronunciación aymara la [u] cerca de un pos-velar [q] suena como [o].
- El sonido de la [l] y la [ll] es muy similar.
- De las reglas de la pronunciación aymara la [i] cerca de un pos-velar [q] suena como [e].

Como se puede ver ambas palabras, aunque escritas de diferente manera, tienen una pronunciación muy similar, esto nos lleva a definir las reglas de igualdad.

DEFINICIÓN. Una regla de reemplazo tomará un carácter de una cadena y utilizará indistintamente cualquiera de sus caracteres de reemplazo, siempre y cuando cumplan con las condiciones de reemplazo, para la comparación con otro carácter.

Por lo expuesto, a continuación se presenta una tabla con las reglas de reemplazo de caracteres (véase el anexo A).

Tabla 3

Reglas de reemplazo para algunos caracteres

	Carácter	Reemplazo	Condición de Reemplazo
1	a	a ä	
2	ä	a ä	
3	i	i ï	
4	o	u ü	
5	c	ss	al lado de la e, i
6	c	k k' kh q q' qh'	al lado de la a o u

3.2.2. PREPROCESAMIENTO DE PATRÓN

El patrón de búsqueda, como se había definido anteriormente en el problema, puede tener errores tipográficos mecánicos y lingüísticos. Los errores mecánicos serán tomados en cuenta al momento de calcular la distancia de edición, mientras que los errores lingüísticos harán corresponder a un carácter sus caracteres de reemplazo al momento de la comparación.

Nótese que en el alfabeto único (véase la sección 2.1.6.1.1) existen grafemas que utilizan para su representación más de un carácter, más aun, comparten un prefijo común con otros grafemas (e.g. “ch”, “chh”, “ch'”), y no existen dos grafemas que concatenados den como resultado otro grafema.

Para poder expresar una cadena en el alfabeto de la implementación, primeramente debemos convertir la cadena en una secuencia de grafemas propios del alfabeto. Al momento de realizar el ajuste de grafemas se aplicará el criterio de “**el grafema con mayor longitud primero**”, lo que quiere decir que primeramente se buscaran entre los grafemas con mayor longitud. Por ejemplo considere que el patrón de búsqueda p = “qhawa”, si no aplicaremos el criterio mencionado, podríamos terminar con una secuencia de grafemas de la siguiente manera [q][h][a][w][a], sin embargo lo correcto sería [qh][a][w][a].

Si bien el anterior procedimiento intentará ajustar los grafemas, como podemos usar más de un carácter para expresar un grafema, pueden existir casos donde los caracteres de un grafema hayan sido escritos mal, por esta razón se incluirán en el alfabeto a todos los caracteres individuales de los grafemas.

3.2.3. DISTANCIA DE EDICIÓN MODIFICADA

La Figura 1 muestra la matriz de programación dinámica resultante después de aplicar la recurrencia (4) para el cálculo de la distancia de edición entre las palabras “COLQUE” y “QULLQI”. Como se puede observar al hallar la distancia de edición con la recurrencia (4) no se toman en cuenta los errores lingüísticos y el resultado no ayuda a resolver el problema planteado.

Figura 1. Matriz de programación dinámica para el cálculo de la distancia de edición entre “QULLQI” y “COLQUE”.

	-	C	O	L	Q	U	E
-	0	1	2	3	4	5	6
Q	1	1	2	3	4	5	6
U	2	2	2	3	4	5	6
L	3	3	3	2	3	4	5
L	4	3	4	3	3	3	4
Q	5	4	3	4	4	4	4
I	6	5	4	4	5	5	5

Extenderemos la distancia de edición para poder resolver el problema planteado, primeramente como se había observado en la sección anterior, el patrón de búsqueda necesita un preprocesamiento antes de poder ser utilizado en el algoritmo para hallar la distancia de edición con otra palabra.

De la ecuación (4) podemos ver que el algoritmo no sufrirá un gran cambio, únicamente se modificará la función de comparación entre caracteres. En lugar de comparar el carácter i-ésimo del patrón de búsqueda con el carácter j-ésimo de la cadena del diccionario, se

compararán los caracteres de reemplazo, si es que cumplen las condiciones de reemplazo, del carácter i-ésimo del patrón de búsqueda con el j-ésimo carácter de la palabra del diccionario.

Figura 2. Matriz de programación dinámica para el cálculo de la distancia de edición modificada entre “QULLQI” y “COLQE”.

	-	C	O	L	Q	E
-	0	1	2	3	4	5
Q	1	0	1	2	3	4
U	2	1	0	1	2	3
LL	3	2	1	0	1	2
Q	4	3	2	1	0	1
I	5	4	3	2	1	0

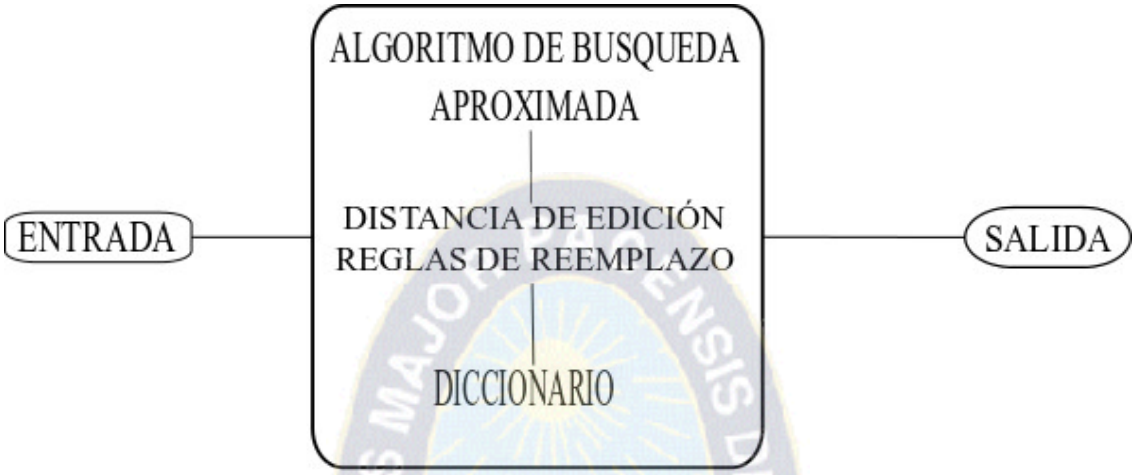
3.3. ESPECIFICACIÓN DEL ALGORITMO

La entrada para el algoritmo de búsqueda aproximada ABCR2 es el patrón de búsqueda, un valor máximo admitido para la distancia k, la salida es un subconjunto de cadenas del diccionario cuya distancia de edición extendida sea a lo más k.

3.4. DISEÑO DEL ALGORITMO

La estructura general del algoritmo de búsqueda aproximada ABCR2 nos permite realizar, primeramente la generación de reglas de reemplazo para el patrón de búsqueda y posteriormente el cálculo de la distancia de edición con las cadenas del diccionario.

Figura 3. Esquema general del algoritmo ABCR2.



3.5. ANÁLISIS DEL ALGORITMO

Como se ha expuesto en la sección 2.3.2.5, el cálculo de la distancia de edición entre las cadenas p y s, utiliza un tiempo y memoria proporcional a $O(|p| \cdot |s|)$.

El único cambio realizado durante el cálculo de la distancia de edición es la inclusión de las reglas de reemplazo en la comparación de caracteres. No es difícil ver que no existen muchas reglas de reemplazo y el reemplazo de un carácter por otro demora un tiempo constante, por lo cual con la nueva comparación de caracteres no se altera el tiempo de ejecución.

Pero si queremos ser rigurosos en el análisis del algoritmo, sea **r** la cantidad máxima de reglas de reemplazo que un carácter cualquiera de nuestro alfabeto posee, entonces el algoritmo para el cálculo de la distancia de edición modificado que se presenta, utiliza tiempo y memoria proporcional a $O(r \cdot |p| \cdot |s|)$.

3.6. IMPLEMENTACIÓN DEL ALGORITMO

Para realizar las pruebas y observar los tiempos de ejecución es necesario implementar el algoritmo de búsqueda aproximada en lenguas indígenas originarias campesinas.

La implementación del algoritmo se realizó en el lenguaje de programación Java, a continuación se describirá el proceso como se implementó el programa.

3.6.1. CLASES

3.6.1.1. REEMPLAZO

Esta clase define la regla de reemplazo, tiene como atributos básicos tres cadenas:

- Reemplazo: El valor con el que debe ser reemplazado el carácter al que pertenece esta regla de reemplazo.
- Antes: La condición de reemplazo que debe cumplir el carácter anterior al que pertenece esta regla en la cadena.
- Después: La condición de reemplazo que debe cumplir el carácter posterior al que pertenece esta regla en la cadena.

3.6.1.2. CARACTER

Esta clase tiene como atributos:

- Caracter (Tipo String): tiene el valor del carácter.
- Reemplazos (Tipo Vector<Reemplazo>): contiene todas las reglas de reemplazo del carácter.

3.6.1.3. ALFABETO

Esta clase solo tiene un atributo, caracteres (Tipo Vector<Caracter>) donde se encuentran todos los caracteres del alfabeto, además de algunos extra para la consideración de los errores ortográficos y de transcripción, además de las reglas de reemplazo para los caracteres.

3.6.1.4. CADENA

Esta clase solo tiene un atributo, cadena (Tipo Vector<Caracter>) que guarda los caracteres de la cadena.

3.6.1.5. CADENAUTIL

Esta clase nos permite transformar una cadena de tipo String a un objeto de tipo Cadena.

3.6.1.6. DICCIONARIO

Esta clase tiene como único atributo palabras (Tipo Vector<Cadenas>), donde se encuentran las palabras del diccionario en tipo Cadena. Durante la instanciación del diccionario se realiza la lectura del archivo donde se encuentran las palabras.

3.6.1.7. ALGORITMO

Esta clase es la más importante del programa, en esta clase se encuentra el método **distanciaDeEdicion** que recibe los parámetros p y s (ambos de tipo Cadena) y realiza el cálculo la distancia de edición entre las cadenas p y s utilizando programación dinámica.

función **distanciaDeEdicion**

```
1.  public static int distanciaDeEdicion(Cadena p, Cadena s) {
2.    int n = p.getCadena().size();
3.    int m = s.getCadena().size();
4.
5.    int dp[][] = new int[n+1][m+1];
6.
7.    for (int i = 0 ; i <= n ; i++) {
8.      for (int j = 0 ; j <= m ; j++) {
9.        int res = oo;
10.       if(i == 0 && j == 0) {
11.         res = 0;
12.       }
13.       if(i > 0) {
14.         res = Math.min(res, 1 + dp[i-1][j]);
15.       }
16.       if(j > 0) {
17.         res = Math.min(res, 1 + dp[i][j-1]);
18.       }
19.       if(i > 0 && j > 0) {
20.         Caracter caracterIzq = p.getCadena().get(i-1);
21.         Caracter caracterDer = s.getCadena().get(j-1);
22.         if(caracterIzq.puedeIgualar(p, i-1, caracterDer)) {
23.           res = Math.min(res, dp[i-1][j-1]);
24.         } else {
25.           res = Math.min(res, dp[i-1][j-1] + 1);
26.         }
27.       }
28.       dp[i][j] = res;
29.     }
30.   }
31.   return dp[n][m];
32. }
```

Fragmento de código 1. Cálculo de la distancia de edición

3.6.1.8. RESULTADO

Esta clase es utilizada unicamente para guardar dos cosas, por una lado la Cadena con la que se compara el patrón de búsqueda y por otro lado la distancia de edición:

- cadena (Tipo Cadena).
- distancia (Tipo entero).

La clase es utilizada con el propósito de poder realizar el ordenamiento de los resultados en base a la distancia de edición.

3.6.2. PROGRAMA PRINCIPAL

En el programa principal se hace uso de los componentes anteriormente desarrollados y la interacción entre ellos se facilita gracias a la programación orientada a objetos.

La variable “patron” y la variable “k” son datos entrada para el programa principal, ambos fueron determinados en la especificación del algoritmo.

Programa Principal

```
1.  Cadena cadenaPatron = CadenaUtil.toCadena(patrón);
2.  Diccionario diccionario = new Diccionario();
3.  Vector<Resultado> resultados = new Vector<Resultado>();
4.  for(Cadena palabra : diccionario.getPalabras()) {
5.    int distancia = Algoritmo.distanciaDeEdicion(cadenaPatron, palabra);
6.    if(distancia <= k) {
7.      resultados.add(new Resultado(palabra, distancia));
8.    }
9.  }
10. Collections.sort(resultados);
11. for(Resultado resultado : resultados.subList(0, 10)) {
12.   System.out.println(resultado.getCadena()+" "+resultado.getDistancia());
13. }
```

Fragmento de código 2. Programa principal

CAPÍTULO IV

4. EVALUACIÓN EXPERIMENTAL

4.1. PRUEBAS

Para las pruebas del algoritmo ABCR2 se utilizó dos diccionarios:

- Diccionario Aymara – Castellano, KAMISARAKI, Saturnino Callo Ticona.
- Diccionario Bilingüe: Aymara – Castellano, Félix Layme Pairumani.

ambos digitalizados para su uso en SIMIDIC (véase el anexo C), a partir de ahora nombraremos a los diccionarios A y B respectivamente. De los diccionarios se extrajeron únicamente las palabras en aymara, aunque se aplicó primeramente un proceso de limpiado pues existían palabras repetidas y algunos símbolos no propios del alfabeto, las palabras fueron guardadas en los archivos **callots.txt** y **laymepf.txt** respectivamente.

De los diccionarios podemos obtener los siguientes datos:

Tabla 4
Datos de los diccionarios A y B

Concepto	Notación	Diccionario A	Diccionario B
Número de cadenas del diccionario	N	10015	6902
Tamaño promedio de las cadenas en el diccionario	λ	8	7
Tamaño máximo de las cadenas del diccionario	λ_m	25	20

También como otro dato relevante, se presenta en la siguiente tabla la frecuencia de palabras en base a la cantidad de caracteres.

Tabla 5

Frecuencia de palabras por cantidad de caracteres de los diccionarios.

Tamaño	**Diccionario A**	**Diccionario B**
1	17	16
2	28	17
3	65	58
4	471	416
5	796	721
6	1189	1139
7	1226	1179
8	1463	1128
9	1445	882
10	1285	672
11	940	356
12	536	177
13	288	92
14	140	31
15	63	11
16	32	3
17	9	2
18	9	1
19	8	0
20	1	1
21	1	0
22	1	0
23	1	0
24	1	0
25	1	0

Se realizó la búsqueda de palabras utilizadas en el castellano cuyo origen es el aymara, es decir, palabras castellanizadas del aymara. El listado de palabras se presenta en el anexo B, para las pruebas en el archivo **castellanizadas.txt** se guardan por cada palabra del anexo B la palabra en aymara escrita correctamente y su respectiva castellanización.

4.1.1. PRECISIÓN DE LOS RESULTADOS DE LA BÚSQUEDA

Para poder medir la precisión de los resultados de la búsqueda, tomaremos como parámetro la posición de la palabra original en la lista de resultados, si es que se encuentra. Consideremos el siguiente cuadro para medir la precisión del resultado de la búsqueda:

Tabla 6

Precisión del resultado de la búsqueda en base a la posición de la palabra.

Precisión	Posición de la palabra original
Excelente	1
Muy buena	2 - 5
Buena	6 - 10
Regular	11 – 15
Mala	> 15
Muy mala	No se encuentra

4.1.2. COMPARACIÓN ENTRE LAS DISTANCIAS DE EDICIÓN CON PALABRAS CASTELLANIZADAS

Para mostrar la diferencia que existe cuando utilizamos la distancia de edición modificada y la no modificada para la comparación entre palabras de las lenguas indígenas originarias campesinas se utilizó las palabras del archivo **castellanizadas.txt**. Se toma como primer parámetro la palabra en aymara correctamente escrita y como segundo parámetro su respectiva castellanización. Se suman las distancias de edición obtenidas en la comparación de dichas palabras y se saca un promedio.

4.1.3. PRUEBA DE PRECISIÓN CON PALABRAS CASTELLANIZADAS

Con esta prueba se intenta medir el grado de precisión del algoritmo de búsqueda aproximada ABCR2 para lenguas indígenas originarias campesinas. Para medir el grado de precisión en la prueba se utilizarán las palabras del archivo **castellanizadas.txt**, se ingresará como patrón de búsqueda la palabra castellanizada y se realizará la búsqueda en el diccionario A, se anotará la posición en que el algoritmo de búsqueda aproximada sitúa a la palabra en aymara correctamente escrita, k representa el valor máximo permitido para la distancia de edición.

4.1.4. PRUEBA DE TOLERANCIA A LOS ERRORES MECÁNICOS

Con esta prueba se intenta medir el grado de precisión con el que el algoritmo ABCR2 responde antes errores mecánicos en las palabras. Para esta prueba se utilizarán las palabras del diccionario A y escogiendo aleatoriamente una posición en la palabra se cometerá un error en su escritura, la palabra con error será el patrón de búsqueda y se anotará la precisión del resultado de búsqueda.

Las substituciones serán generadas tomando aleatoriamente un carácter de la palabra y substituyendolo por un carácter contiguo en un teclado qwerty, para así simular un error en la escritura de esa palabra por causa de la presión de una tecla contigua a la deseada. Las palabras generadas son guardadas en el archivo **substituciones.txt**.

Las inserciones serán generadas tomando aleatoriamente un carácter de la palabra e insertando un carácter contiguo en un teclado qwerty, para así simular un error en la escritura de esa palabra por causa de la presión de dos teclas. Las palabras generadas son guardadas en el archivo **inserciones.txt**.

Los borrados serán generados tomando aleatoriamente un carácter de la palabra y borrandolo, para así simular el olvido de la presión de una tecla. Las palabras generadas son guardadas en el archivo **borrados.txt**.

4.1.5. PRUEBA DE TOLERANCIA A LOS ERRORES LINGÜÍSTICOS

Con esta prueba se intenta medir el grado de precisión con el que el algoritmo ABCR2 responde antes errores lingüísticos en las palabras. Para esta prueba se escogieron 10 palabras del diccionario A y por cada palabra se utilizarán las reglas de reemplazo para generar palabras con errores lingüísticos, las palabras con errores lingüísticos serán los patrones de búsqueda y se anotará la precisión de los resultados de las búsquedas. La palabra original y las palabras generadas son guardadas en el archivo **errores-linguísticos.txt**. En total se generan 8388 palabras con errores lingüísticos.

4.1.6. PRUEBAS CON LA DISTANCIA DE FRECUENCIA

Como se había definido la distancia de frecuencia entre dos palabras, se debe contar la frecuencia individual de cada carácter y realizar la comparación entre los vectores de frecuencia respectivos. El problema con esta prueba es que se intenta no penalizar los errores de tipo lingüístico y como hacemos uso de las reglas de reemplazo en caracteres al momento de contar un carácter entramos en una posible ambigüedad.

Por la razón antes expuesta, se omite esta prueba.

4.2. RESULTADOS

4.2.1. COMPARACIÓN ENTRE LAS DISTANCIAS DE EDICIÓN CON PALABRAS CASTELLANIZADAS

En la siguiente tabla se muestra los resultados arrojados por la distancia de edición no modificada y la modificada entre la castellanización y la palabra escrita correctamente, para una muestra de 57 palabras.

Tabla 7
Resultados obtenidos en la comparación entre distancias de edición con palabras castellanizadas.

	Distancia de edición	
	No modificada	Modificada
Suma de las distancias	148	28
Promedio	2.6	0.5

De la Tabla 7 podemos ver la distancia de edición entre una palabra en aymara correctamente escrita y su castellanización con el algoritmo de distancia de edición modificada en promedio reduce en 2 la distancia con respecto al algoritmo convencional.

4.2.2. PRECISIÓN CON PALABRAS CASTELLANIZADAS

En la tabla 8 están los resultados obtenidos de la prueba de precisión para el algoritmo ABCR2, como se puede observar el 68% de los resultados son de precisión excelente. En la tabla 9 están los resultados obtenidos de la prueba de precisión para el algoritmo convencional, como se puede a lo más 18% de los resultados son de precisión excelente. Al comparar los resultados de las tablas 8 y 9 podemos observar que existe una diferencia de 50% en los resultados de precisión excelentes.

Tabla 8

Resultados obtenidos de la prueba de precisión con palabras castellanizadas para el algoritmo ABCR2 con un valor limite k para la distancia de edicion.

Precisión	k = 1	k = 2	k = 3
Excelente	39	39	39
Muy buena	12	14	14
Buena	0	0	0
Regular	0	0	0
Mala	0	3	4
Muy mala	6	1	0
Total	57	57	57

Tabla 9

Resultados obtenidos de la prueba de precisión para el algoritmo convencional con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	5	9	10
Muy buena	5	14	18
Buena	1	3	5
Regular	0	3	4
Mala	0	4	7
Muy mala	46	24	13
Total	57	57	57

4.2.3. TOLERANCIA A LOS ERRORES MECÁNICOS

4.2.3.1. TOLERANCIA A LAS SUBSTITUCIONES

En las tablas 10 y 11 se muestra los resultados obtenidos en la prueba de tolerancia a las substituciones, el algoritmo convencional tiene un 86% de resultados de excelente precisión, en cambio el algoritmo ABCR2 tiene un 63%.

Tabla 10

Resultados obtenidos de la prueba de tolerancia a las substituciones para el algoritmo ABCR2 con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	6012	6384	6384
Muy buena	2049	2324	2324
Buena	489	605	605
Regular	191	250	250
Mala	286	467	468
Muy mala	1004	1	0
Total	10031	10031	10031

Tabla 11

Resultados obtenidos de la prueba de tolerancia a las substituciones para el algoritmo convencional con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	8656	8656	8656
Muy buena	1225	1225	1225
Buena	116	116	116
Regular	24	24	24
Mala	10	10	10
Muy mala	0	0	0
Total	10031	10031	10031

4.2.3.2. TOLERANCIA A LOS BORRADOS

En las tablas 12 y 13 se muestra los resultados obtenidos en la prueba de tolerancia a los borrados, el algoritmo convencional tiene un 59% de resultados de excelente precisión, en cambio el algoritmo ABCR2 tiene un 43%.

Tabla 12

Resultados obtenidos de la prueba de tolerancia a los borrados para el algoritmo ABCR2 con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	4305	4400	4400
Muy buena	3291	3335	3335
Buena	904	917	917
Regular	350	354	354
Mala	461	561	562
Muy mala	257	1	0
Total	9568	9568	9568

Tabla 13

Resultados obtenidos de la prueba de tolerancia a los borrados para el algoritmo convencional con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	5993	5993	5993
Muy buena	3095	3095	3095
Buena	334	334	334
Regular	106	106	106
Mala	40	40	40
Muy mala	0	0	0
Total	9568	9568	9568

4.2.3.3. TOLERANCIA A LAS INSERCIONES

En las tablas 14 y 15 se muestra los resultados obtenidos en la prueba de tolerancia a las inserciones, el algoritmo convencional tiene un 86% de resultados de excelente precisión, en cambio el algoritmo ABCR2 tiene un 59%.

Tabla 14

Resultados obtenidos de la prueba de tolerancia a las inserciones para el algoritmo ABCR2 con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	5936	6292	6293
Muy buena	2132	2426	2426
Buena	471	601	601
Regular	183	239	239
Mala	301	472	472
Muy mala	1008	1	0
Total	10031	10031	10031

Tabla 15

Resultados obtenidos de la prueba de tolerancia a las inserciones para el algoritmo convencional con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	8610	8610	8610
Muy buena	1274	1274	1274
Buena	111	111	111
Regular	26	26	26
Mala	10	10	10
Muy mala	0	0	0
Total	10031	10031	10031

4.2.4. TOLERANCIA A LOS ERRORES LINGÜÍSTICOS

En las tablas 16 y 17 se muestra los resultados obtenidos en la prueba de tolerancia a los errores lingüísticos, el algoritmo ABCR2 tiene un 78% de resultados de precisión excelente mientras que el algoritmo convencional apenas alcanza un 6% dando una diferencia de 72%.

Tabla 16

Resultados obtenidos de la prueba de tolerancia a los errores lingüísticos para el algoritmo ABCR2 con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	6576	6576	6576
Muy buena	1792	1792	1792
Buena	4	4	4
Regular	4	4	4
Mala	12	12	12
Muy mala	0	0	0
Total	8388	8388	8388

Tabla 17

Resultados obtenidos de la prueba de tolerancia a los errores lingüísticos para el algoritmo convencional con un valor limite k para la distancia de edición.

Precisión	k = 1	k = 2	k = 3
Excelente	64	219	515
Muy buena	16	92	303
Buena	4	31	82
Regular	0	8	47
Mala	1	15	124
Muy mala	8303	8023	7317
Total	8388	8388	8388

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

El presente trabajo propone el algoritmo de búsqueda aproximada ABCR2 en lenguas indígenas originarias campesinas basado en la distancia de edición modificada, mediante la inclusión de reglas de reemplazo para la comparación de cadenas y/o caracteres.

Con el desarrollo del algoritmo ABCR2 se ha utilizado un modelo de error que toma en cuenta los errores en la escritura de las lenguas indígenas originarias campesinas, dicho modelo se basa en las reglas de reemplazo (véase la sección 3.2.1) para la comparación entre caracteres del alfabeto.

En la sección 3.3 se definen las especificaciones del algoritmo ABCR2, tanto los valores de entrada como de salida.

Definimos la función de distancia de edición modificada (véase la sección 3.2.3) como nuestro criterio principal de comparación entre cadenas para las lenguas indígenas originarias campesinas.

Comparando la distancia de edición modificada con la no modificada con palabras castellanizadas, comprobamos que la distancia de edición modificada en promedio reduce en 2 la distancia en la búsqueda de palabras castellanizadas, lo que produce un resultado más preciso en la búsqueda aproximada de palabras castellanizadas.

Se ha observado al comparar los resultados de la prueba de precisión con palabras castellanizadas (véase las tablas 8 y 9) con el algoritmo ABCR2 y el algoritmo convencional, que el algoritmo ABCR2 produce un 50% más de resultados de precisión excelente.

En la comparación de resultados del algoritmo ABCR2 y el algoritmo convencional en la prueba de tolerancia a errores de mecánicos (véase las tablas 10 - 15) se ha visto que el algoritmo convencional produce un 77% de resultados de precisión excelente comparado con un 55% producido con el algoritmo ABCR2, es decir, el algoritmo convencional produce un 22% más de resultados excelentes.

En la comparación de resultados del algoritmo ABCR2 y el algoritmo convencional en la prueba de tolerancia a errores de lingüísticos (véase las tablas 16 y 17) se ha visto que el algoritmo ABCR2 produce un 72% de resultados de precisión excelente más que el algoritmo convencional.

En conclusión de los resultados de evaluación experimental tenemos que el algoritmo de búsqueda aproximada ABCR2 mejora la precisión de los resultados de la búsqueda de palabras castellanizadas y palabras con errores lingüísticos, aunque no se puede decir lo mismo de las palabras que tienen errores mecánicos.

5.2. RECOMENDACIONES

El presente trabajo utilizó como lengua base el aymara, si se desea experimentar con las demás lenguas indígenas originarias campesinas se deberían extender las reglas de reemplazo (véase la sección 3.2.1) y verificar que el algoritmo ABCR2 funciona de igual manera.

En la distancia de edición modificada se utilizan las reglas de reemplazo descritas en la sección 3.2.1, se espera que existan otras propuestas para la distancia de edición.

En la sección de pruebas se omitió la prueba con la distancia de frecuencia debido a una posible ambigüedad, se espera la definición de un criterio para realizar el conteo de la frecuencia y así poder utilizar la distancia de frecuencia como un filtro para la búsqueda aproximada.

La búsqueda aproximada es un campo de estudio bastante amplio, existen métodos más sofisticados para ello como: los árboles de prefijos, la generación de vecindarios, pivotes en espacios métricos, particionamiento de patrones, etc. El presente trabajo propone el algoritmo ABCR2 que es en comparación bastante simple, pero es una base para la extensión a los demás métodos. Si se desea mejorar la eficiencia se puede implementar alguno de los métodos ya mencionados.

Se espera que futuros trabajos de investigación estudien los métodos presentados en el presente trabajo más a fondo y los apliquen a las demás lenguas indígenas originarias campesinas, implementando un algoritmo específico para cada una de ellas.

BIBLIOGRAFÍA

- Apaza Ignacio (2012). LA DESCOLONIZACIÓN CULTURAL, LINGÜÍSTICA Y EDUCATIVA EN BOLIVIA. Estudios Bolivianos.
- Bertonio, L. (2011). Transcripción del Vocabulario de la lengua aymara. La Paz, Bolivia: Radio San Gabriel, Instituto de Lenguas y Literaturas Andinas-Amazónicas.
- Bolivia (2009), Constitución Politica del Estado Plurinacional de Bolivia.
- Boytsov, L. (2011). Indexing methods for approximate dictionary searching. J. Exp. Algorithmics JEA Journal of Experimental Algorithmics,16 (1).
- Saca (2011). Corrector ortográfico del idioma aymara con automatas finitos, Tesis de Grado, Universidad Mayor de San Andrés, Carrera de Informática.
- Tavosanis, M. (2007). A causal classification of orthography errors in web texts. IJCAI-2007: Workshop on Analytics for Noisy Unstructured Text Data, 99-106.

BIBLIOGRAFÍA WEB

- ChartsBin statistics collector team 2011, Number of Endangered Languages by Country, ChartsBin.com, Recuperado 4 November, 2015 a partir de http://chartsbin.com/view/1339.
- Endangered Languages Project. (s. f.). Recuperado 4 de noviembre de 2015, a partir de http://www.endangeredlanguages.com/about/
- Heggarty, P. (2006). Quechua Pronunciation and Spelling. Recuperado el 18 de enero de 2016, de http://www.quechua.org.uk
- Lenguas de Bolivia. (2015, octubre 22). En Wikipedia, la enciclopedia libre. Recuperado a partir de https://es.wikipedia.org/w/index.php?title=Lenguas_de_Bolivia&oldid=86024935
- Pre Cumbre reflexiona sobre problemas que atraviesa la comunicación indígena en el país. (s.f.). Recuperado 4 de noviembre de 2015, a partir de http://www.boliviaentusmanos.com/noticias/bolivia/122746/pre-cumbre-reflexiona-sobre-problemas-que-atraviesa-la-comunicacion-indigena-en-el-pais.html.
- Projects | United Nations Educational, Scientific and Cultural Organization. (s. f.). Recuperado 4 de noviembre de 2015, a partir de http://www.unesco.org/new/en/culture/themes/endangered-languages/projects/

ANEXOS

ANEXO A: LISTA COMPLETA DE REGLAS DE REEMPLAZO

	Carácter	Reemplazo	Condición de Reemplazo
1	a	a ä	
2	ää	a ä	
3	ii	i ï	
4	ïï	i ï	
5	e	i ï	
6	ë	i ï	
7	u	u ü	
8	ü	u ü	
9	o	u ü	
10	ö	u ü	
11	p	p ph p'	
12	ph	p ph p'	
13	p'	p ph p'	
14	t	t th t'	
15	th	t th t'	
16	t'	t th t'	
17	j	j x	
18	x	j x	
19	l	l ll	
20	ll	l ll y	
21	y	ll y	
22	r	r	
23	m	m n ñ	
24	n	m n ñ	
25	ñ	m n ñ	
26	c	s	delante de la i o e
27	c	k k' kh q q' qh	delante de la a o u
28	g	j x	Al lado de la i o e

29	s	s ch chh	
30	sh	s ch chh	
31	ch	s ch chh	
32	chh	s ch chh	
33	k	k k' kh q q' qh	
34	kh	k k' kh q q' qh	
35	k'	k k' kh q q' qh	
36	q	k k' kh q q' qh	
37	qh	k k' kh q q' qh	
38	q'	k k' kh q q' qh	
39	ch	ch ch' chh	
40	ch'	ch ch' chh	
41	chh	ch ch' chh	
42	b	w	
43	v	w	
44	z	s	

ANEXO B: LISTA DE PALABRAS AYMARA CASTELLANIZADAS

	Escritura correcta	Castellanización
1	qullqi	colque
2	ch'arki	charque
3	ch'arkikanka	charquecan
4	ch'uñu	chuño
5	luqutu	locoto
6	chhuxllu	choclo
7	awina	avena
8	laranja	naranja
9	luxma	lucuma
10	phurut'i	poroto
11	tunasa	tuna
12	turasno	durazno
13	uwa	uva
14	yuka	yuca
15	tampu	tambo
16	awayu	aguayo
17	mitala	metal
18	iqiqu	ekeko
19	kuku	cucu
20	achuqalla	achocalla
21	pinkillu	pinquillo
22	qina	quena
23	llamiru	llamerada
24	wayñu	huayño
25	alwa	alba
26	allpaqa	alpaca
27	inka	inca
28	aymara	aimara
29	ayni	aine

30	aysa	asia
31	kamanchaka	camanchaca
32	kukuli	cucule
33	pisitunga	pisitunka
34	sapallu	zapayo
35	wisk'acha	viscacha
36	ayramp'u	airampo
37	allqamari	alcamar
38	aru	aro
39	k'allana	callana
40	kallapu	callapo
41	wanaku	guanaco
42	ch'uspa	chuspa
43	qulla	colla
44	quchala	cochala
45	wayra	guaira
46	warapo	guarapo
47	mut'i	mote
48	juminta	huminta
49	phalt'a	palta
50	ch'illkha	chilca
51	khirkinchu	quirquincho
52	suruxchi	sorojchi
53	tutura	totora
54	ulluku	olluco
55	wikuña	vicuña
56	wincha	vincha
57	kumpari	compadre

ANEXO C: IMPLEMENTACIÓN

SimiDic es un diccionario electrónico código libre y abierto para dispositivos móviles Android. Esta aplicación fue creada para facilitar la comunicación entre personas de diferentes culturas y mantener el uso de lenguas nativas con ayuda de los teléfonos móviles. Incluye varios diccionarios de lenguas nativas americanas, incluyendo aymara, quechua, guaraní y mapuche.

Este diccionario esta disponible bajo la licencia **GPL 3.0 o superior**, cuenta con más de 10000 instalaciones.

Se realizó la implementación del algoritmo ABCR2 en el SimiDic, si bien el diccionario cuenta con la existencia de multiples lenguas el algoritmo ABCR2 se hace disponible para la lengua aymara.

La Paz, 4 de abril de 2016

Señores:

HONORABLE CONSEJO DE CARRERA

CARRERA INFORMÁTICA

FAC. CIENCIAS PURAS Y NATURALES

UNIVERSIDAD MAYOR DE SAN ANDRES

Presente:

Ref: AVAL PARA LA DEFENSA DE TESIS DE GRADO

De mi mayor consideración:

Mediante la presente, me dirijo a ustedes, en calidad de **Tutor Metodológico** para informar que luego de haber realizado el seguimiento de la Tesis de Grado titulada: "**DISEÑO DE ALGORITMO DE BUSQUEDA APROXIMADA EN LENGUAS INDIGENAS ORIGINARIAS CAMPESINAS**" presentado por el Universitario **JOSE CARLOS LAURA RAMIREZ** con C.I. **6839033 LP.**, para optar al título de Licenciatura en Informática, Mención Ciencias de la Computación.

En este sentido, presento mi **conformidad y aval** respectivo para la defensa pública de la Tesis de Grado de acuerdo a Reglamento vigente en la Universidad Mayor de San Andrés.

Sin otro particular, me suscribo con las atenciones más distinguidas.

M.Sc. Edgar Palmiro Clavijo Cardenas

TUTOR METODOLOGICO

c.c. Arch

La Paz, 4 de abril de 2016

Señor
M.Sc. Edgar Palmiro Clavijo Cardenas
TUTOR METODOLOGICO
Presente

Ref: CONFORMIDAD Y AVAL DE TESIS DE GRADO

De mi mayor consideración:

Tengo a bien dirigirme a su persona, para darle a conocer que luego de efectuar el seguimiento a la estructura y contenido de la Tesis de Grado titulado: "**DISEÑO DE ALGORITMO DE BUSQUEDA APROXIMADA EN LENGUAS INDIGENAS ORIGINARIAS CAMPESINAS**", elaborado por el Universitario: **JOSE CARLOS LAURA RAMIREZ**, con C.I. **6839033 LP**, en calidad de **ASESOR** expreso mi conformidad con el contenido y la forma de trabajo, dando **mi aval**, para que el postulante pueda realizar la defensa de la Tesis de Grado, para optar al título de Licenciado en Informática mención Ciencias de la Computación, de acuerdo a normas y reglamento vigentes.

Sin otro particular, me suscribo de su persona con las consideraciones más distinguidas.

M.Sc. Jorge Humberto Terán Pomier
ASESOR

c.c. Arch.