UNIVERSIDAD MAYOR DE SAN ANDRES
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMATICA

TESIS DE GRADO

"CORRECTOR ORTOGRAFICO DEL IDIOMA AYMARA CON AUTOMATAS FINITOS"

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMATICA
MENCION: INGENIERIA DE SISTEMAS INFORMATICOS

POSTULANTE: EDWIN NESTOR SACA APAZA
TUTOR METODOLOGICO: LIC. GROVER ALEX RODRIGUEZ RAMIREZ
ASESOR: M.Sc. ALDO RAMIRO VALDEZ ALVARADO

LA PAZ – BOLIVIA
2014

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.

TODOS LOS DERECHOS RESERVADOS. EL USO NO AUTORIZADO DE LOS CONTENIDOS PUBLICADOS EN ESTE SITIO DERIVARA EN EL INICIO DE ACCIONES LEGALES CONTEMPLADOS EN LA LEY DE DERECHOS DE AUTOR.

AGRADECIMIENTOS

Quiero agradecer a Dios, a mi padre Felipe Saca Huanca y a mi madre Filomena Apaza de Saca por haber permitido que yo tenga educación universitaria dentro de la Universidad Mayor de San Andres.

También me gustaría agradecerle a la Sociedad Científica Estudiantil de la Facultad de Ciencias Puras y Naturales, el cual me ha apoyado durante mis dos semestres de investigación.

RESUMEN

La gramática ofrece reglas para el empleo de letras y signos de la escritura. La forma correcta para la escritura en cualquier idioma es vital ya que puede producir malas interpretaciones o mensaje erróneos.

El idioma aymara es el lenguaje oral del Mundo Andino con el cual se comunican y no hay correctores para estas comunidades.

El alfabeto del idioma aymara, está compuesto por 30 fonemas segméntales: 26 consonantes, 3 vocales (/a/, /i/, /u/) y un alargamiento vocálico (//). Los fonemas consonánticos a su vez se dividen en sordos y sonoros.

La presente tesis de grado es propuesta con el propósito de mejorar la gramatica escrita en lenguaje aymara utilizando teoría de autómatas. Para la verificación de las palabras. Por medio del corrector se establecerá una escritura adecuada entre el usuario y el sistema. El desarrollo del proceso de corrección automática de una palabra.

La teoría de autómatas verifica las palabras para poder corregir correctamente en la lengua aymara, y tiene los siguientes pasos: descomposición de la palabra, clasificación, corregir y mostrar las alternativas para la corrección, una vez realizado estos pasos se desarrolló el autómata.

Índice de General

AGRADECIMIENTOS **2**

RESUMEN **3**

1. MARCO REFERENCIAL **12**

1.1. Introducción . . . 12
1.2. Antecedentes . . . 13
1.3. Problema . . . 14
1.3.1. Planteamiento del problema . . . 14
1.3.2. Formulación del Problema . . . 15
1.4. Objetivos . . . 15
1.4.1. Objetivo General . . . 15
1.4.2. Objetivos Específicos . . . 15
1.5. Hipótesis . . . 16
1.5.1. Identificación de variables . . . 16
1.6. Justificación . . . 16
1.6.1. Científica . . . 16
1.6.2. Social . . . 17
1.6.3. Técnica . . . 17
1.6.4. Económica . . . 17
1.7. Alcances y Limites . . . 18
1.7.1. Alcances . . . 18
1.7.2. Limites . . . 18

2. MARCO TEÓRICO **19**
2.1. Corrector Ortográfico 19
2.1.1. Ortografía 19
2.1.2. Consonantismo 20
2.1.3. Vocalismo 21
2.1.4. Descripción 22
2.1.5. Correctores ortográficos 23
2.1.6. Correctores Gramaticales 23
2.1.7. Aspell 23
2.2. Teoría de autómatas 24
2.2.1. Definiciones básicas 28
2.2.2. Autómatas finitos 30
2.2.2.1. Autómata finito deterministico 30
2.2.2.2. Representación como diagramas de transiciones . . . 31
2.2.3. Autómatas finitos no deterministicos 32
2.3. FOMA 34
2.3.1. Lanzamiento de foma 34
2.3.2. Compilando expresiones regulares 35
2.3.3. Examinar FSMs 35
2.3.4. Visualización gráfica 36
2.3.5. La pila 37
2.3.6. Símbolos especiales y Unicode 37
2.3.7. Impresión de la palabras que un FSM acepta 37
2.3.8. Prueba de palabras 38
2.3.9. Símbolos de varios caracteres en foma 38
2.3.10. Los símbolos esenciales? y 0 en expresiones regulares 39
2.3.11. El símbolo @ en especial FSMs 39
2.3.12. El comando definir 40
2.4. La lengua del idioma Aymara 41
2.4.1. Gramática de la lengua Aymara 42
2.4.2. Símbolos Fonéticos 42
2.4.3. Los principales elementos gramaticales 44

2.4.4. Símbolos ortográficos 45
2.4.5. Organización y escritura de sufijos 47
2.5. Metodologías 53
2.5.1. Introducción 53
2.5.2. Metodos de Acceso métrico 53
2.5.3. Estructuras de Indexación y Búsqueda 54
2.5.3.1. Tipos de Consultas 54
2.5.3.2. Árboles para Funciones Discretas 55
2.5.3.3. Árboles para Funciones de Distancia Continua 55
2.5.4. Construcción del Slim- Tree 57
2.5.4.1. Métodos del ChooseSubTree (Elegir Sub Árbol) . . . 57
2.5.4.2. Algoritmos de Partición del Slim- Tree 58
2.5.5. Metodologia para Correctores Ortográficos 58
2.5.5.1. Método de prefijos o sufijos 58
2.5.5.2. Distancia Levenshtein 59
2.5.6. Morfología de Estados Finitos 60
2.5.7. Morfologia en Software Libre 61
2.5.8. El Procesamiento del Lenguaje Natural 62
2.5.9. Características morfológicas del Castellano 64
2.5.9.1. Derivaciones de género y número 65
2.5.9.2. Conjugación verbal 65
2.5.9.3. Formas enclíticas 66
2.5.9.4. Nombres derivados de verbos 66
2.5.9.5. Adverbios derivados de adjetivos 66
2.5.9.6. Superlativos y diminutivos 66
2.5.9.7. Vocales acentuadas 66
2.5.10. Implementación del modelo 67
2.5.10.1. Derivaciones de género y número 67
2.5.10.2. Formas enclíticas 68
2.5.10.3. Nombres derivados de verbos 68
2.5.10.4. Adverbios derivativos de adjetivos 68
2.5.10.5. Superlativos y diminutivos 69

2.5.11. Generación del Diccionario 69
2.6. Distribución t de Student 71
2.6.1. Teoría de pequeñas muestras 71

3. DISEÑO METODOLÓGICO 73
3.1. Introducción 73
3.1.1. Planificación Lingüística 74
3.1.1.1. Ligüística Formal 74
3.1.2. Método de prefijos o sufijos 74
3.1.2.1. Clasificación morfológica del Aymara 75
3.1.2.2. Derivaciones número 75
3.1.2.3. Conjugación verbal 75
3.1.2.4. Formas enclíticas 76
3.1.2.5. Alargamiento Vocalica 76
3.1.2.6. Elisión Vocalica 76
3.2. Componentes del Corrector 76
3.3. Variables del modelo 80
3.4. Descripción formal del Autómata 80
3.4.1. Función de la división de la palabra 80
3.4.2. Función de análisis de comparación 81
3.4.3. Corrección 81
3.5. Infraestructura 85
3.5.1. Software 85
3.5.2. Hardware 85
3.5.3. Funciones del software 85
3.5.4. Foma ejecutando el corrector Ortográfico 87

4. EVALUACIÓN DE RESULTADOS 89
4.1. Análisis de datos y resultados 89
4.2. Datos obtenidos del corrector ortográfico 91
4.3. Datos obtenidos del lingüista 91

5. CONCLUSIONES Y RECOMENDACIONES **94**
5.1. Conclusiones 94
5.2. Recomendaciones 95

Bibliografía **96**

A. Textos para la corrección **97**
A.1. TIWULANPI HUALLATAMPI 97
A.2. TIWULAMPIT WAYRAMPITA 98
A.3. KUNA ASKIRAK 98
A.4. PÁXPÁKUBT KÚSILUJNTI 99

B. Tabla t-student **100**

C. Documentación **102**

Índice de Figuras

2.1. Diagrama de transiciones de el hombre, el lobo, la cabra y la col. [9, página 34] . . . 27
2.2. Ejemplo de un Autómata. [6, Apuntes] . . . 32
2.3. Ejemplo de Autómata No Deterministico. [6, Apuntes] . . . 33
2.4. Foma corriendo en gnu-linux. [2, Enlace] . . . 35
2.5. Probando regex en foma. [2, Enlace] . . . 35
2.6. Examinando FSM en foma. [2, Enlace] . . . 36
2.7. Gráfico de foma. [2, Enlace] . . . 36
2.8. Gráfico impresión. [2, Enlace] . . . 37
2.9. Gráfico ejemplo de impresión. [2, Enlace] . . . 38
2.10. Gráfico ejemplo de prueba. [2, Enlace] . . . 38
2.11. Gráfico de @. [2, Enlace] . . . 40
2.12. Gráfico de define. [2, Enlace] . . . 40
2.13. Corrector Ortografico. [5, página 6] . . . 59
2.14. Estructura Morfológica simplificada del castellano. [8, página 3] . . . 65
2.15. Verbos Regulares e Irregulares. [8, página 5] . . . 68
2.16. Generación del Diccionario. [8, página 7] . . . 70

3.1. Estructura Morfológica del Aymara [Elaboración Propia] . . . 75
3.2. Esquema general del corrector ortográfico. [Elaboración Propia] . . . 78
3.3. Composición de un Sustantivo. [Elaboración Propia] . . . 79
3.4. Armado de la palabra. [Elaboración Propia] . . . 81
3.5. Autómata de Verbo. [Elaboración Propia] . . . 83
3.6. Autómata de Sustantivo. [Elaboración Propia] . . . 84

3.7. Archivo arsunakawa02.lex [Elaboración Propia] 86
3.8. Archivo varianteteclado.foma [Elaboración Propia] 86
3.9. Archivo ayamara_corrector.foma [Elaboración Propia] 87
3.10. Ejecución del Corrector. [Elaboración Propia] 88
3.11. Corrección de la palabra auquili. [Elaboración Propia] 88

Índice de Tablas

2.1. Autómata No Deterministico. [6, Apuntes] 34
2.2. Consonantes de Aymara [1, página 16] 43
2.3. Vocales del Aymara [1, página 16] 43
2.4. Grafias Consonánticas [1, página 18] 46
2.5. Grafías vocálicas [1, página 18] 46
2.6. Sufijos derivativos o lexicos nominales [1, página 19] 48
2.7. Sufijos derivativos o léxicos verbales [1, página 21] 52

3.1. Reglas de la palabra. [Elaboración Propia] 79

4.1. Diccionario de datos del corrector [Elaboración propia] 91
4.2. Diccionario de datos del lingüista Aymara[Elaboración propia] 91
4.3. Datos unidos del lingüista y el corrector [Elaboración propia] 92

Capítulo 1

MARCO REFERENCIAL

1.1. Introducción

La gramática normativa ofrece reglas para el empleado de letras y signos de la escritura.

La forma correcta para la escritura en cualquier idioma es sumamente importante, por que induce a errores de interpretación provocando malas interpretaciones en la comunidad.

La implementación del corrector ortográfico en el lenguaje aymara mediante los autómatas, permitirá una correcta interpretación de esta lengua para los hablantes y no hablantes dentro del Estado Plurinacional de Bolivia.

La comunidad Aymara dentro del Estado Plurinacional de Bolivia, no tiene un Corrector Ortográfico en este idioma para ver la escritura correcta, ya que existe mas de 1.415.679 millones que se auto identifican como Aymaras en el ultimo censo llevado el 2012, se ve necesario construir este corrector para las personas que hablan y escriben en este idioma, ya que las persona auto identificadas están dispersas en su mayoría en La Paz, Oruro y Potosi y en su menoría en los otros departamentos diversificados a nivel Bolivia.

La implementación del Corrector Ortográfico del Idioma Aymara mediante autómatas, Lenguajes Formales, FOMA, FSM. Considerando la parte de clasificación de palabras y el orden correcto de los sufijos.

El corrector se presenta con el objetivo de ayudar a las personas hablantes en el uso correcto de la escritura del Idioma Aymara.

El Idioma Aymara ha sido un lenguaje oral del Hombre Andino y no fue representado en forma escrita, pero existen trabajos como el Diccionario Aymara-Español de Felix Layme, Diccionario Aymara-Español de Felix Layme que son una base en la forma correcta de la escritura Aymara.

Para esta tesis se ve necesario tener conocimientos del Idioma Aymara, para identificar las palabras y dividirlos en sustantivo, verbo, adverbio, adjetivo. Una vez dividido se da el orden de que sufijos vienen después de las palabras mencionadas, ya que el Idioma Aymara es netamente sufijante.

1.2. Antecedentes

El lenguaje natural es la lengua o idioma hablado escrito por humanos para propósitos generales de comunicación. Son aquellas lenguas que han sido generadas espontáneamente en un grupo de hablantes con propósito de comunicarse, a diferencia de otras lenguas, como puedan ser una lengua construida, los lenguajes de programación o los lenguajes formales usados en el estudio de la lógica formal, especialmente la lógica matemática.

El término lenguaje convencional se refiere al estudio de las propiedades computacionales y de otro tipo implicadas en la comprensión, producción y uso de las lenguas naturales como el Idioma Aymara.

Existen trabajos de distintos autores en el que se detalla la forma correcta de escribir el Idioma Aymara, como la propuesta del Peruano Cerrón Palomino, y tam-

bién existen trabajos de los Bolivianos Felix Layme, Teofilo Layme, el que más se destaca es el trabajo de Felix Layme con su diccionario Diccionario Bilingüe Aymara Castellano – Castellano Aymara, al no seguir reglas convencionales como los otros idiomas se usara FOMA, con el cual se puede preparar reglas personalizadas para el Idioma Aymara.

Foma está basado en FSM (Finete State Morphology), que ha sido creado por el Estado Unidense Mans Hulden el cual ha sido construido en lenguaje c para manejar la estructura de las palabras.

Existen experiencias como del idioma Euskera en España que hizo su corrector con foma un proyecto creado por Mans Hulden University of the Basque Country.

En la carrera de Informática de la Universidad Mayor de San Andrés se desarrollaron dos trabajos similares y un trabajo realizado en España University of the Basque Country:

- Interprete Español - Quechua de Marleni Ibone Vedia Rodriguez, cuyo objetivo principal es la interpretación semántica del Idioma Quechua del territorio Boliviano [7, página 1].
- Traducción semántica y sintáctica del Idioma Español al Aymara utilizando autómatas de Teodosia Paye Ticona, cuyo objetivo es la traducción del Idioma Español al Aymara dentro de la ciudad de La Paz dentro del territorio Boliviano [9, página 1].
- Corrector Ortográfico del Euskera, realizado en España, con el cual están revalorizando este idioma dentro de la comunidad Española.

1.3. Problema

1.3.1. Planteamiento del problema

- La corrección de las palabras de la escritura Aymara se hace manualmente con la guía de expertos o diccionarios lo cual ocasiona perdida de tiempo.

- La escritura de los hablantes en el idioma aymara no es correcta porque no hay políticas del Gobierno para la enseñanza de este idioma o es poco difundido.
- Las personas que no saben escribir o tienen errores en su escritura no pueden corregir por falta de un Corrector Ortográfico.
- Como no existe un corrector las personas hablantes hacen el uso de diccionarios y reglas que se encuentran en los diccionarios bilingües del idioma Aymara y en su gramática.
- Los diccionarios bilingües no están al alcance de todos los que escriben y hablan esta lengua porque hay pocas publicaciones que detallen y sintetizan la misma.
- Los expertos de esta lengua son contados y por esta razón el contratar a los mismos tiene un costo alto.

1.3.2. Formulación del Problema

Todos los puntos mencionados anteriormente muestran la falta de un Corrector Ortográfico del Idioma Aymara.

¿Como se puede corregir palabras del Idioma Aymara, considerando la clasificación de las mismas y el orden de sufijos?

1.4. Objetivos

1.4.1. Objetivo General

Plantear un modelo de Corrector Ortográfico con autómata finito que permita la detección de palabras escritas en el Idioma Aymara, para corregir los errores introducidos por el usuario el cual mostrara diferentes opciones de las posibles soluciones de la palabra correcta.

1.4.2. Objetivos Específicos

- Enseñar la escritura correcta del Idioma Aymara.

- Establecer las reglas del orden de los sufijos de la lengua aymara.
- Construir autómatas para el análisis morfológico para la verificación de las palabras o posibles correcciones de las mismas.
- Implementar el diccionario de datos para el corrector ortográfico.

1.5. Hipótesis

H_i: "El corrector ortográfico con autómatas finitos usando **FOMA** permite corregir las palabras con errores ortográficos, del idioma Aymara con un reconocimiento de palabras de hasta un 70 % ".

H_O: "El corrector ortográfico con autómatas finitos usando **FOMA** no permite corregir las palabras con errores ortográficos, del idioma Aymara con un reconocimiento de palabras de hasta un 70 % ".

1.5.1. Identificación de variables

Variable Independiente (VI): Corrector ortográfico del Idioma Aymara.

Variable Dependiente (VD): Corrección de palabras.

Variable Autómata (VA): Teoría de autómatas con morfologia de estados finitos.

1.6. Justificación

1.6.1. Científica

Con la implementación de autómatas en FOMA es capaz de generar grandes cantidades de palabras clasificadas dando alternativas de palabras de corrección, nos brindara una herramienta para la corrección o propuestas de palabras para corregir

un determinado texto, articulo y otros.

El hecho de implementar autómatas en el corrector ortográfico permitirá observar la interacción entre usuario y el corrector para el desarrollo del software para la corrección automática permitirá corregir los errores ortográficos de palabras escritas en el idioma aymara para hablantes y no hablantes de este idioma, y con esto se apoyara a la comunidad Aymara dentro de Bolivia.

1.6.2. Social

La presente investigación a desarrollar se justifica socialmente, porque beneficiara a los hablantes y no hablantes del idioma Aymara ya que podrán utilizar el corrector para corregir sus errores ortográficos dentro del Estado Plurinacional de Bolivia, quienes serán los encargados de interactuar con el corrector.

Esta investigación ayudara a resolver el problema de la escritura, en hablar para las personas que saben y no saben hablar el idioma aymara.

1.6.3. Técnica

Con la implementación de autómatas en la corrección de errores en la escritura de aymara, nos brindara una herramienta para la corrección de las palabras. El hecho de implementar autómatas finitos en un entorno como el foma, nos permitirá observar la interacción de orden de sufijos y palabras en la escritura aymara.

La presente investigación es una propuesta con el propósito de mejorar la corrección de la escritura de la lengua aymara.

1.6.4. Económica

La clasificación de palabras y la corrección se hace manualmente provocando perdida de tiempo y la contratación de un experto para la corrección ya que no existe una herramienta para corregir.

Los usuarios del corrector ortográfico dispondrán de tiempo para hacer sus distintas actividades reduciendo el costo de usar diccionarios o la contratación de especialistas para la corrección.

En la actualidad, se busca mejorar el gasto de recursos económicos por parte de las Instituciones Privadas y el Estado Plurinacional de Bolivia.

1.7. Alcances y Limites

1.7.1. Alcances

Se hace con foma la codificación para los casos mencionados para posteriormente jalar de un Lex las raíces para combinar con sufijos de la lengua aymara.

Se hace el reconocimiento del aymara de Bolivia en sus variantes en los Gobiernos Departamentales de La Paz, Oruro y Potosi, dentro de las provincias de La Paz la forma de escribir y la forma de hablar es diferente como en Omasuyus, Los andes, Pacajes y otros.

El presente trabajo se realiza mediante un autómata que permitirá la corrección de Sustantivos, verbos, adjetivos y adverbios en la lengua aymara, el cual ayudara a la forma correcta de escribir.

Se construirá un prototipo en FOMA para la corrección de palabras clasificadas en Sustantivos, verbos, adjetivos y adverbios del idioma Aymara.

1.7.2. Limites

Entre los limites del estudio, es que la herramienta no se realizara para un procesador de texto como ser Libre Office, Open Office, Microsoft Office y otros.

No considera las posibles variantes de las regiones de Perú y otros países donde se habla el Idioma Aymara.

Capítulo 2

MARCO TEÓRICO

2.1. Corrector Ortográfico

Un corrector ortográfico en la informática, es una aplicación de software que se utiliza para analizar textos con el fin de detectar de forma automática o manual y corregir faltas ortográficas.

2.1.1. Ortografía

La ortografía de acuerdo con Enrique Fontanillo (1986), es la rama de la gramática que se ocupa de las letras del alfabeto y de sus reglas combinatorias para transcribir correctamente los sonidos de una lengua dada. Así mismo para Theodor Lewandowski (1995), la ortografía es la norma de la escritura, una teoría de la corrección en la escritura, la normalización de la representación gráfica de los signos lingüísticos, por tanto, un sistema de reglas que asegura la constancia y uniformidad de la escritura.

La ortografía para Guillermo Castelao (2000), es la guía de la escritura que presentan principios básicos, de los cuales mencionamos los más relevantes: El uso del sentido común , el adquirir un rol comunicativo, el error ortográfico por el conocimiento a medias o mal de los significados de las palabra, y por ultimo, la forma de la palabra como un acuerdo cultural que excede el plano de las relaciones personales.

El diccionario denominado Oxford Advanced Learnerś Dictionary define los conceptos acerca de la ortografía, acentuación, puntuación y las grafías de la siguiente manera: "**Ortografía**, el sistema de ortografía de un idioma; **acento**, el énfasis dado a un sílaba o palabra por medio de la tensión o de paso; **estrés**, una fuerza extra utilizado al pronunciar una palabra o una sílaba en particular; a **puntuar**, para dividir a escribir en oraciones, frases, etc por utilizando marcas especiales, por ejemplo, puntos y aparte, signos de interrogación, etc; Textbf gráfico, arte visual, por ejemplo, la impresión, la escritura o dibujo."

Todos ellos así tal conceptualización tiene que ver directamente con la escritura fónica o logográfica.

En tal sentido, clasificamos a nuestro entender en tres partes: la acentuación, los signos de puntuación y la grafemática. Estas tres ramas están transversalizadas por las consonantes y las vocales; los lexemas y los sufijos, además de los constituyentes del enunciado u oración.

2.1.2. Consonantismo

Según Fontanillo se refiere al sistema organizado de las consonantes de una lengua dada. Este autor también establece la precedencia de las consonante oclusivas respecto a las fricativas, las orales se producen antes que las nasales y las labiales antes que las velares, dentales o palantales.

En tal sentido, los oclusivos quedan en la parte superior de cualquier cuadro fonético y fonológico, incluyendo así las combinaciones de fonemas compuestos y africados.

Akmajian y otros (1987) refiriéndose a las consonantes oclusivas del inglés y del castellano señalan que la */t/ y /d/* se diferencian en ambos idiomas en el lugar de articulación, así en el inglés son alveolares y en el español, dentales. Asimismo, ellos demuestran la presencia de consonantes glotales */?/ y /h/* del inglés y la ausencia

de las mismas en el castellano, aunque en este idioma (español) existe solo la velar */x/* en lugar de la glotal, por ejemplo en las palabras *<teja>/texa/,<julio>/xulyo/*, entre otras.

De acuerdo a Cerrón Palomino (1994), el consonantismo del aymara consta de 26 segmentos que se clasifican en base a dos criterios articuladores: a) punto de formación y b) manera de producción. El autor explica que según el primer criterio distingue seis órdenes de consonantes: bilabiales, dento alveolares, palatales, velares, postvelares y glotales. Así mismo tomando en cuenta el segundo criterio, son cinco clases de consonantes; oclusivas, fricativas, liquidas y semiconsonantes.

Sabemos que las consonantes juegan un papel importante en la contrastación de signos lingüísticos (palabras con significado) autónomo, ello es razón suficiente para desarrollar el uso de las consonantes en la ortografía aymara.

2.1.3. Vocalismo

Tomando en cuenta la identificación de Enrique Fontanillo, vocalismo es el sistema vocálico de una lengua, familia o grupo de lenguas, En tal sentido, el aymara es un idioma que se compone según su estudio fonológico por un vocalismo y un consonantismo.

Precisamente, para definir las vocales, citamos a Jean Dubois quien en su Diccionario de Lingüística explica lo siguiente: “Las vocales son fonemas que presentan el rasgo vocálico sin poseer ningún rasgo consonántico. Son sonidos musicales debidos a las vibraciones periódicas del aire laríngeo que pasa libremente a través del canal bucal”. Así mismo Juana Gil Fernandez (1999) al respecto indica: “La vocal seria en principio el sonido más audible, el más abierto, sin obstáculo alguno a la salida del aire y articulado con menor esfuerzo que la consonante”.

De acuerdo a Adrián Akmajian y otros (1987) el sistema vocálico varia de idioma a idioma, así el español presenta vocales */a„e,i,o,u/* que se caracterizan por su localización en anteriores y posteriores y por su grado de abertura (o densidad),

en vocales altas (o difusas), medias y bajas (acústicamente densas). A su vez estos autores distingue en el inglés dos clases principales de vocales caracterizadas como breves frente a largas, o flojas frente a tensas, respectivamente. Las vocales breves descritas del inglés americano son /I, ε, æ, $\bigcup$, $\wedge$,a ,$\supset$/.

También ellos desarrollan las vocales reducidas $\dot{+}$,

Así mismo las vocales largas (tensas) y diptongos las agrupan en diptongos formados por vocal + glide externa, con valor fonémico distintivo, que son */iy, ey, ay, oy, uw, ow, aw/*. Así como vemos el inglés distingue fonémicamente mayor número de vocales breves que el castellano.

Según Marths Hardman y otros (1984), así como varios lingüistas, afirman la existencia de solo tres fonemas vocálicos en el aymara: */i/, /a/, y /u/*. Sin embargo, en el plano fonético estas vocales pueden distribuirse en varios alófonos vocoides, pero que de ninguna manera afectan en el significado o contenido del signo lingüistico.[1, página 11,12,13,14]

2.1.4. Descripción

No debe confundirse la ortografía con la gramática. El corrector ortográfico compara las palabras del texto con las palabras en un diccionario. Si las palabras del texto se encuentran en el diccionario, estas son aceptadas, de lo contrario, el corrector propone términos similares. La gramática, por otro lado, comprueba que las palabras del texto (existentes en el diccionario), cumplen ciertas normas de la gramática (orden de las palabras, etc.) y de la semántica (frases que tengan sentido, etc.). Para la corrección de errores gramaticales se pueden utilizar un corrector gramatical, si bien hoy en día muchos correctores ortográficos también cumplen esta función, así como errores en la puntuación del texto.

Un corrector ortográfico **no puede garantizar que un texto sea correcto**, pues, aunque las palabras en una oración existan, pueden que la palabra no tenga la semántica correcta. Ejemplo "*La baca es un animal que da leche*" sería aceptado porque la palabra baca existe. Por lo tanto, el corrector ortográfico no detecta el

error ortográfico (“baca” en lugar de “vaca”).

Además, suelen tener problemas al detectar palabras prefijadas o con elementos compositivos, o incluso con vocablos sufijados, dando como opciones “correctas” unas formas que precisamente no son correctas ortográficamente [**?** , página 1].

2.1.5. Correctores ortográficos

Los correctores ortográficos mas conocidos en el mundo son los siguientes:

- **Aspell** del proyecto GNU;
- Entre los motores de búsqueda en internet: Google o Yahoo!.

2.1.6. Correctores Gramaticales

Los correctores gramaticales más conocidos en el mundo son los que se detalla a continuación:

- LanguageTool (Open Source)
- Microsoft Word
- Stilus - Corrector ortográfico, gramatical y de estilo de Daedalus.

2.1.7. Aspell

Gnu Aspell, normalmente llamada sólo Aspell, es un software libre corrector ortográfico diseñado para sustituir ispell. Es el corrector ortográfico estándar para el GNU sistema de software. También compila para otros Unix-como sistemas operativos y de Windows. El programa principal está licenciado bajo la Licencia Pública General de GNU (GNU LGPL), la documentación bajo la Licencia de Documentación Libre GNU (GNU FDL). Diccionarios para ello están disponibles por cerca de 70 idiomas. El mantenedor principal es Kevin Atkinson.

2.2. Teoría de autómatas

Es una rama de las ciencias de la computación que estudia las máquinas abstractas y los problemas que éstas son capaces de resolver. La teoría de autómatas está estrechamente relacionada con la teoría del lenguaje formal ya que los autómatas está estrechamente relacionado con la teoría del lenguaje formal ya que los autómatas son clasificados a menudo por la clase de lenguajes formales que son capaces de reconocer.

Un autómata finito es un modelo matemático de un sistema, con entradas y salidas discretas. El sistema puede estar en cualquiera de un momento finito de configuraciones o "estados". El estado del sistema resume la información concerniente a entradas anteriores y que es necesaria para determinar el comportamiento del sistema para entradas posteriores. El mecanismo de control de un elevador es un buen ejemplo de un sistema de estados finitos. El mecanismo no recuerda todas las demandas previas de servicio, sino sólo el piso en el que se encuentra la dirección de movimiento (hacia arriba o hacia abajo) y el conjunto de demandas de servicio aún no satisfechas.

En la ciencia de la computación encontramos muchos ejemplos de sistemas de estados finitos, y la teoría de autómatas finitos es una útil herramienta para el diseño de tales sistemas. Un primer ejemplo es un circuito de interrupción, como la unidad de control de una computadora. Un circuito de interrupción está compuesto por un número finito de compuertas, cada una de las cuales puede estar en una de dos condiciones, por lo general denotadas por 0 y 1. Estas condiciones podrían, en términos electrónicos, ser dos niveles de voltaje distintos en la compuerta de salida. El estado de una red de interrupción con n compuertas es, por lo tanto, una de las 2^n asignaciones de 0 y 1 para las diferentes compuertas. Aunque el voltaje en cada compuerta puede tomar cualquiera de un conjunto infinito de valores, el circuito electrónico está diseñado de tal manera que sólo los dos voltajes correspondientes a 0 y 1 son estables, y otros voltajes se ajustarán, casi inmediatamente, a uno de esos voltajes. Los circuitos de interrupción se diseñan intencionalmente de esta manera, de modo que puedan ser vistos como sistemas de estado finito, separando, por ello, el diseño lógico de una computadora de la implementación electrónica.

Ciertos programas, utilizados con frecuencia, como los editores de texto y los analizadores de léxico que se encuentran en la mayoría de los compiladores, son diseñados como sistemas de estado finito. Por ejemplo, una analizador de léxico recorre los símbolos de un programa de computación para localizar cadenas de caracteres correspondientes a identificadores, constantes numéricas, palabras reservadas, etc. En este proceso el analizador de léxico necesita recordar solamente una cantidad finita de información, como qué longitud de un prefijo de una palabra reservada ha visto desde el inicio. La teoría de los autómatas finitos se utiliza mucho en el diseño de procesadores eficientes de cadenas de este tipo y de otros.

La computadora misma puede ser vista como un sistema de estado finito, aunque hacer esto no resulta tan útil como uno quisiera, Teóricamente, el estado del procesador central, memoria principal y almacenamiento auxiliar en cualquier instante de tiempo es uno de una gran número finito de estados. Estamos suponiendo, por supuesto, que existe un número fijo de discos, tambores, cintas, etc., disponibles, y que uno no puede extender la memoria indefidamente. Tomar un computadora como un sistema de estado finito, sin embargo no es realista o satisfactorio matemáticamente. Establece un límite artificial a la capacidad de memoria. Y, por tanto, no es capaz de capturar la esencia real de la computación. Para captar de manera apropiada el concepto de computación se necesita una memoria potencialmente infinita.

Resulta tentador, también, considerar el cerebro humano como un sistema de estado finito. El número de células cerebrales o neuronales es limitado, cuando mucho 2^{35}. Podemos considerar, aunque existe evidencia de lo contrario, que el estado de cada neurona puede describirse mediante un pequeño número de bits. Si esto es así, la teoría del estado finito se aplica al cerebro. Sin embargo, el número de estados es tan grande que es poco potable que este planteamiento tenga como resultado observaciones útiles sobre el cerebro humano, no más que las suposiciones de estado finito ayudan a entender sistemas de computación grandes por finitos.

Tal vez la razón más importante para estudiar los sistemas de estado finito es la neutralidad del concepto, como lo indica el hecho de que surge en diferentes y múl-

tiples lugares. Esto es una indicación de que hemos captado el concepto de un tipo fundamental de sistemas, un tipo que es rico en estructura y aplicaciones potenciales.

Un ejemplo Antes de definir de manera formal los sistemas de estado finito, considerados un ejemplo. En la orilla izquierda de un río se encuentra un hombre, junto con un lobo, una cabra y un col. Hay un bote con la capacidad suficiente para llevar al hombre y a uno de los tres. El hombre con la col y demás compañeros deben cruzar el río, y el hombre puede llevar a uno solo a la vez. Sin embargo, si el hombre deja solos al lobo y a la cabra en cualquier lado del río, con toda seguridad que el lobo se comerá a la cabra. Del mismo modo, si la cabra y la col se quedan juntas, la cabra se comerá a la col. ¿Es posible que se pueda cruzar el río sin que nada sea comido por nadie?.

El problema se puede modelar observando que la información pertinente está en los ocupantes de cada orilla después de cruzar. Existen 16 subconjuntos del hombre (H), el lobo (L), la cabra (G) y la col (C). Un estado corresponde al subconjunto que se encuentra en la ribera izquierda. Los estados están etiquetados por parajes separadas por un guión, por ejemplo HG-LC, en donde los símbolos que están a la izquierda del guión denotan el subconjunto que está en la orilla izquierda del río; los símbolos que están a la derecha del guión representan el subconjunto que se encuentran en la orilla opuesta. Algunos de los 16 estados, como GC-HL, son fatales y nunca deben accesarse al sistema.

Las “entradas” al sistema son las acciones del hombre. Puede cruzar solo (entrada h), con el lobo (entrada l), la cabra (entrada g) o con la col (entrada c). El estado inicial es HLGC-0 y el estado final es 0-HLGC. En la Figura 2.1 se muestra el diagrama de transición.

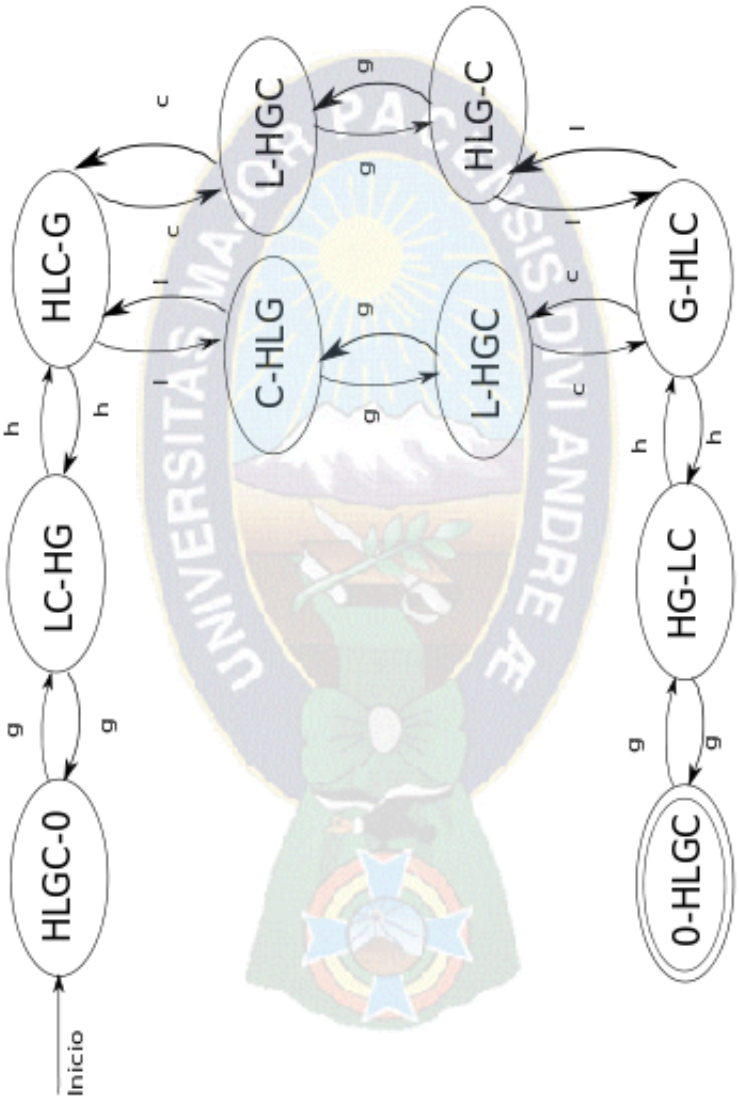

Figura 2.1: Diagrama de transiciones de el hombre, el lobo, la cabra y la col. [9, página 34]

Hay dos soluciones del problema igualmente cortas, como puede verse si se buscan las trayectorias que van del estado inicial al final (que está dentro de un círculo doble). Existen un número infinitamente grande de soluciones diferentes del problema, pero todas, con excepción de dos, involucran ciclos inútiles. El sistema de estado finito puede ser considerado como el que define a un lenguaje infinito, el conjunto de todas las cadenas que etiquetan las trayectorias que van del estado inicial al final.

2.2.1. Definiciones básicas

Un autómata finito (FA) consisten en un conjunto finito de estados y un conjunto de transiciones de estado a estado, que se dan sobre símbolos de entrada tomados de un alfabeto *sum*. Para cada símbolo de entrada existe exactamente una transición a partir de cada estado (posiblemente de regreso al mismo estado). Un estado, por lo general denotado como q_0, es el estado inicial, en que el autómata comienza. Algunos estados están designados como final o de aceptación.

A un (FA) se le asocia un grafo dirigido, conocido como diagrama de transiciones de la manera siguiente. Los vértices del grafo corresponden a los estados del FA. Si existe una transición del estado q al p sobre la entrada a, entonces existe un arco con etiqueta a que va del estado q al p, en el diagrama de transiciones. El FA acepta una cadena x si la secuencia de transiciones correspondientes a los símbolos de x conduce el estado inicial aun estado de aceptación.

Para describir formalmente el comportamiento de un FA sobre una cadena, necesitamos extender la función de transición δ para que se pueda aplicar a un estado y una cadena, más que a un estado y un símbolo. Definimos una función $\hat{\delta}$ de $Q \times \sum *$.La intención es que $\hat{\delta}_{(q,w)}$ represente al estado en el que estará el FA después de leer la cadena w del estado q. Dicho de otra forma, $\hat{\delta}_{(q,w)}$ es el único estado p tal que existe una trayectoria en el diagrama de transiciones de q a p, con la etiqueta w. De manera formal, definimos.

1. Es $\hat{\delta}_{(q,\in)} = q, y$

2. Para todas las cadenas w y símbolos de entrada a.

$$\hat{\delta}_{(q,wa)} = \delta(\hat{\delta}_{(q,w)}, a)$$

De esta forma (1) establece que sin leer un símbolo de entrada el FA no puede cambiar de estado, (2) nos dice cómo encontrar el estado después de leer una cadena entrada no de estado, (2) nos dice como encontrar el estado después de leer una cadena no vacía wa. Esto es, hallar el estado $p = \delta(\hat{q, w})$ después de leer w. Y entonces calcular el estado $\delta(p, a)$.

Puesto que $\hat{\delta}(q, a) = \delta(\delta(q, \hat{\in}), a) = \delta(q, a)$ [haciendo $w =\in$ en la regla (2) arriba], no puede haber desacuerdo entre δ y $\hat{\delta}$ sobre los argumentos para los cuales se definieron. En adelante, por razones de conveniencia, utilizaremos δ en lugar de $\hat{\delta}$.

La entrada es leída símbolo por símbolo, hasta que es "consumida" completamente (piense en esta como una cinta con una palabra escrita en ella, que es leída por una cabeza lectora del autómata; la cabeza se mueve a lo largo de la cinta, leyendo un símbolo a la vez) una vez la entrada se ha agotado, el autómata se detiene.

Dependiendo del estado en el que el autómata finaliza se dice que este ha aceptado o rechazado la entrada. Si este termina en el estado"acepta" el autómata acepta la palabra. Si lo hace en el estado rechaza", el autómata rechazó la palabra, el conjunto de todas las palabras aceptadas por el autómata constituyen el lenguaje aceptado por el mismo.

Los conceptos básicos de símbolos, palabras, alfabetos y Clausura Kleene son comunes en la mayoría de las descripciones de los autómatas. Estos son:

Símbolo es un dato arbitrario que tiene algún significado a o efecto en la máquina. A estos símbolos también se les llama "letras" o "átomos".

Palabra es una cadena finita formada por la concatenación de un número de símbolos.

Alfabeto es un conjunto finito de símbolos. Un alfabeto se indica normalmente con S, que es el conjunto de letras en un alfabeto.

Lenguaje es un conjunto de palabras, formado por símbolos en un alfabeto dado. Puede ser infinito.

Clausura de Kleene es un lenguaje se puede considerar como un subconjunto de todas las posibles palabras. El conjunto de todas las palabras puede, a su vez, ser considerado como el conjunto de todas las posibles concatenaciones.

2.2.2. Autómatas finitos

Pueden ser de dos tipos.

- **Autómata finito deterministico**, es aquel que se encuentra en un solo estado. Luego de leer un secuencia de símbolos de entrada.
- **Autómata finito no deterministico**, esto puede estar en varios estados luego de leer una secuencia de símbolos de entrada.

2.2.2.1. Autómata finito deterministico

Convención Nos esforzamos en usar los mismos símbolos para significar la misma cosa a lo largo de todo el material sobre autómatas finitos. En particular, a menos que se establezca de otra manera, el lector podrá suponer que:

El Autómata finito deterministico, es denotado como A, se define como una quíntupla.

$$A = \{Q, \Sigma, \delta, q_0, F\}$$

donde:

Q: Conjunto de estados.

Σ: Es el alfabeto de entrada.

δ: Función de transición $\delta : Q \times \Sigma \to Q$

q_0: Estado inicial.

F: Conjunto de estados finales.

Los autómatas finitos se representan de dos maneras.

- **Diagrama de transiciones**.

2.2.2.2. Representación como diagramas de transiciones

Un diagrama de transiciones para el Autómata finito deterministico AF A.

$$A = \{Q, \Sigma, \delta, q_0, F\}$$

Un grafo es definido de la siguiente manera.

- Existe un nodo por cada estado de Q.
- Para $q \in Q$ y un símbolo a que pertenece el alfabeto $a \in \Sigma$, $\delta(q, a) = p, p \in Q$
- Existirá una arco desde el nodo q al nodo p etiquetados con a.
- Si existe varios símbolos que provoca la misma transición, estos se separan por comas o algún otro separador (/).
- Existe una flecha dirigido al estado inicial, etiqueta con la palabra inicio.
- Los nodos correspondientes a los estados finales o de aceptación presentan doble circulo, las restantes presentan círculos simples .

- Las transiciones se refleja en el mismo grafo.

Ejemplo: Un autómata de un Automata Finito Deterministico en la Figura 2.2.

$$A = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_1\})$$

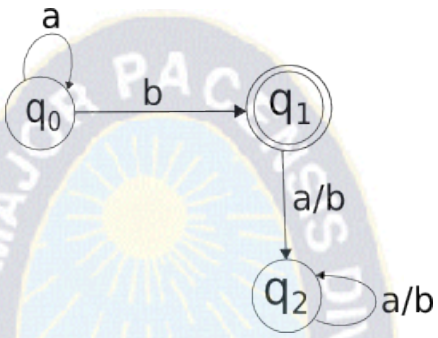

Figura 2.2: Ejemplo de un Autómata. [6, Apuntes]

2.2.3. Autómatas finitos no deterministicos

Introducimos ahora el concepto de autómata finito no determinístico. Se llegará a la conclusión de que cualquier conjunto aceptada por un autómata finito no deterministico puede, también, ser aceptado por una autómata finito determinístico. Sin embargo, el autómata finito no determinístico es un concepto útil cuando se trata de demostrar teoremas. También el concepto de no deterministico juega un papel central de teorías de lenguajes de computación, y es útil para comprender, en un principio y de manera completa, este concepto en un contexto muy simple. Más adelante, encontraremos autómatas cuyas versiones deterministas y no deterministas se sabe que no son equivalentes, y otros para los cuales la equivalencia es una profunda e importante interrogante.

Considérese una modificación al modelo del autómata finito para permitirle ninguna, una o más transiciones de un estado sobre el mismo símbolo de entrada. Este nuevo modelo se conoce como *autómata finito no determinístico (NFA)*.

Son aquellos que tienen la capacidad de estar varios estados simultáneamente , luego de leer un símbolo o cadena.

El autómata finito no deterministico, representado NFA, se define como la quintupla.

$$N = (Q, \sum, \delta, q_0, F)$$

donde cada componente excepto δ, tiene el misma interpretación que en los AF.

La función de transición δ, toma dos argumentos un estado y un símbolo del alfabeto y devuelve como resultado un conjunto de cero o mas estados.

Ejemplo Diseñe un autómata NFA que acepte el lenguaje $L = (w \in \{0,1\} * /w$ terminen en 01 [3, página 15-23] que se detalla en la Figura 2.3.

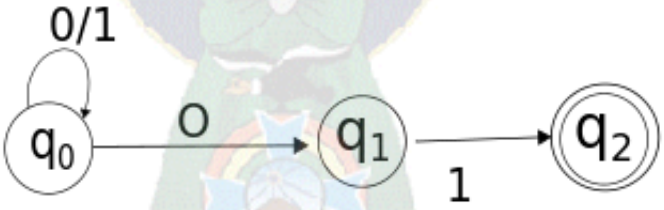

Figura 2.3: Ejemplo de Autómata No Deterministico. [6, Apuntes]

NFA $L = (w \in \{0,1\} * /w$ terminen en 01

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	0	$\{q_2\}$
$*q_2$	0	0

Tabla 2.1: Autómata No Deterministico. [6, Apuntes]

$$N = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, q_2)$$

2.3. FOMA

La interfaz de Foma esta destinado a facilitar la compilación de expresiones regulares en autómatas finitos y traductores (maquina de estados finitos/ FSMs). Las maquinas complejas normalmente se compilan en FSMs especificando a través de una serie de archivos de comandos que contienen las expresiones regulares y las definiciones de los EFM. Una vez FSM se construyen con foma, puede ser almacenado en archivos binarios o de texto y utilizados para otros fines a través de la forma API, la linea de comandos flookup utilidad, u otras herramientas. (FSM finite state machines).

Las maquinas complejas son normalmente compiladas dentro de maquinas de estados finitos, especificando términos en ellos numero de archivos con scripts contenidos. Una vez construido con foma la maquina de estado finito, esto se almacena en binario texto y usado en foma.

2.3.1. Lanzamiento de foma

Foma reside en el path de linux, foma al ejecutarse debería mostrar el siguiente mensaje. El número 0 siguiendo el símbolo indica el número de FSMs actualmente recopila y almacena en una pila normal, que se detalla en la Figura 2.4.

```
mylinux$ foma

Foma, version 0.9.13alpha
Copyright © 2008-2010 Mans Hulden
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; for details, type "help license"

Type "help" to list all commands available.
Type "help <topic>" or help "<operator>" for further help.

foma[0]:
```

Figura 2.4: Foma corriendo en gnu-linux. [2, Enlace]

2.3.2. Compilando expresiones regulares

Las expresiones regulares son compiladas dentro de la maquinas de estados finitos con el comando regex. Por ejemplo. compilará la expresión regular $a * b$ en una

```
foma[0]: regex a b*;
263 bytes. 2 states, 2 arcs, Cyclic.
foma[1]:
```

Figura 2.5: Probando regex en foma. [2, Enlace]

maquina de estado finitos que se añade en la parte superior de la pila (por eso el símbolo del ahora lee: hay un FSM en la pila). tengan en cuenta que las expresiones regulares deben terminar con un punto y coma, que se muestra en la Figura 2.5.

2.3.3. Examinar FSMs

La red de impresión de comandos (o simplemente la red abreviado) se imprimirá información sobre el FSM en la parte superior de la pila, como en la Figura 2.6:

```
foma[1]: net
Sigma: a b
Size: 2.
Net: 6B8B4567
Flags: deterministic pruned minimized epsilon_free
Arity: 1
Ss0:    a -> fs1.
fs1:    b -> fs1.
foma[1]:
```

Figura 2.6: Examinando FSM en foma. [2, Enlace]

Aquí la interfaz de informes de algunas propiedades de la FSM que la expresión regular cedió (ab), el tamaño del alfabeto (2 símbolos), un nombre arbitrario, y las propiedades de la FSM (determinista, podado, minimizado, épsilon gratis). También el hecho de que el FOMA es un autómata de un traductor.

2.3.4. Visualización gráfica

Suponiendo que el software necesario esta instalado (Graphviz / un visualizador de imagen) de autómatas / transductores también se pueden visualizar de forma gráfica a través de la red de vista (o simplemente ver) de comandos, como en la Figura 2.7:

```
foma[1]:ver
```

lo que abrirá una ventan que muestra la FSM en la parte superior de la pila:

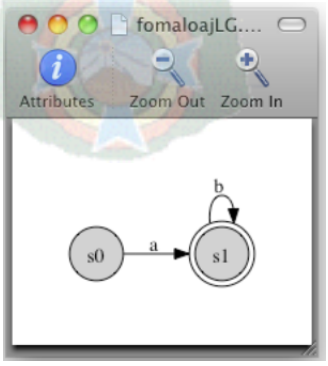

Figura 2.7: Gráfico de foma. [2, Enlace]

2.3.5. La pila

Como se ha mencionado, foma compila expresiones regulares en una pila interna y el símbolo muestra cuantas FSM se encuentran actualmente almacenado. La pila se puede borrar con el comando clear. Remover sólo la máquina superior se hace con el comando pop. La mayoría de los comandos de la interfaz que operan en el FSM (por ejemplo, ver neto) funcionan en la parte superior de la máquina almacenad en la pila.

2.3.6. Símbolos especiales y Unicode

Foma trata a todos de entrada y salida como UTF-8 y no se ocupa de otras codificaciones. Casi todos los símbolos latinos básicos, excepto los símbolos alfanuméricos son especiales y necesitan ser escapado si sus valores literales están destinados. Escapar de un símbolo puede ser hecho por cualquiera de prefijar con un signo de porcentaje, o que encierra el símbolo entre comillas. Por ejemplo, la expresión regula:

```
foma[0]:regex "%"|% $
```

produce un autómata que acepta dos palabras: % y $. Fuera de la gama de Basic Latin Unicode hay escape hay que hacer, con la excepción de unos pocos operadores especiales.

2.3.7. Impresión de la palabras que un FSM acepta

Las palabras de comando de impresión (o sólo palabras) se imprimirán todas las palabras que el FSM superior en la pila acepta, en la Figura 2.8.

```
foma[0]: regex a b c | d e ;
317 bytes. 5 states, 5 arcs, 2 paths.
foma[1]: words
abc
de
foma[1]:
```

Figura 2.8: Gráfico impresión. [2, Enlace]

Si el FSM es cíclico y aceptan un número infinito de palabras, sólo una selección de ellos será impreso. Por ejemplo en la Figura 2.9.

```
foma[0]: regex [a b c | d e]*;
389 bytes. 4 states, 5 arcs, Cyclic.
foma[1]: words

abc
de
foma[1]:
```

Figura 2.9: Gráfico ejemplo de impresión. [2, Enlace]

2.3.8. Prueba de palabras

Los comandos de abajo o hacia arriba (la dirección única que importa para transductores) se moverán foma a un modo prompt especial donde las palabras pueden ser probadas en contra de la FSM en la parte superior de la pila. La tecla CTRL D vuelve al prompt regular. Por ejemplo en la Figura 2.10:

```
foma[0]: regex a b* c;
257 bytes. 3 states, 3 arcs, Cyclic.
foma[1]: down
apply down> a
???
apply down> abbc
abbc
apply down> ^D
foma[1]:
```

Figura 2.10: Gráfico ejemplo de prueba. [2, Enlace]

En el ejemplo anterior, la expresión regular $ab*c$ fue compilado primero. Luego de las palabras de una *abbc* fueron probados contra la FSM, y sólo este último fue aceptada.

2.3.9. Símbolos de varios caracteres en foma

Una fuente frecuente de confusión con las expresiones regulares *foma* permite símbolos de varios caracteres (o símbolos de cuerda) en las expresiones regulares. Esto es muy diferente de los formalismos de expresiones regulares convencionales que sólo permiten a los símbolos de un solo símbolo. En foma, los símbolos deben

estar separados por espacios si son para ser destinados como algo separado. Por ejemplo:

```
regex cat;
```

producirá una FSM que acepta una sola palabra que consiste en el símbolo solo gato. La mayoría de las veces, esto no es deseable, y lo que realmente estaba destinado era:

```
regex cat;
```

que producirá un FSM que acepta sólo una palabra que consta de tres símbolos c, a, t.

También se puede evitar tener que escribir espacios entre los símbolos encerrando las palabras entre llaves, que forma siempre interpretar símbolos separados como unidos, es decir:

```
regex {cat}
```

2.3.10. Los símbolos esenciales? y 0 en expresiones regulares

En las expresiones regulares, el símbolo especial? se refiere a cualquier símbolo sencillo que sea. Por ejemplo:

```
regex? A
```

producirá un FSM que acepte las palabras que constan de cualquier símbolo único, seguido del símbolo a.

El símbolo 0 siempre se corresponde con el símbolo *épsilo* (ϵ), refiriéndose a la cadena vacía. La minúscula símbolo griego épsilon ϵ, puede también ser utilizado para este propósito.

Por ejemplo, la expresión regular.

```
regex ab | 0
```

producirá un FSM que acepta dos cadenas: la cadena ab y la cadena vacía. El símbolo épsilon y la cadena vacía se utiliza con mayor frecuencia con transductores.

2.3.11. El símbolo @ en especial FSMs

En las transiciones de la FSM, el especial de símbolo @ se utiliza para indicar cualquier símbolo fuera del alfabeto de ese FSM. Este símbolo tiene un significado

especial en las expresiones regulares, sin embargo, y no debe ser utilizado a menos que el literal símbolo @ se refería. El símbolo @ a menudo se produce como resultado del uso de cualquier del. En las expresiones regulares. Por ejemplo, la expresión regular.

```
? regex * a *?;
```

se refiere a cualquier palabra que contenga al menos una a-símbolo en alguna parte. Esta expresión regular se compila en el siguiente FSM, en la Figura 2.11:

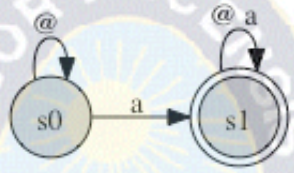

Figura 2.11: Gráfico de @. [2, Enlace]

2.3.12. El comando definir

El comando define puede ser utilizado para etiquetar los FSM. Estas etiquetas pueden ser reutilizados en las expresiones regulares. El formato de comando define es como sigue [2, página 1]:

```
define name regula-expression;
```

Por ejemplo en la Figura 2.12:

```
foma[0]: define Words c a t | d o g | m o u s e;
defined Words: 631 bytes. 10 states, 11 arcs, 3 paths.
foma[0]: regex Words "-" Words;
849 bytes. 20 states, 23 arcs, 9 paths.
foma[1]: words
cat-cat
cat-dog
cat-mouse
dog-cat
dog-dog
dog-mouse
mouse-cat
mouse-dog
mouse-mouse
```

Figura 2.12: Gráfico de define. [2, Enlace]

2.4. La lengua del idioma Aymara

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 se habla en nor-oeste y la parte nor-central de Bolivia (sobre todo en la ciudad de La Paz, la zona lacustre del lago Titicaca y la mayor parte del Altiplano), en el sur de Perú (en los departamentos de Puno Tacna, Moquegua y Arequipa y toda la zona lacustre), en el norte de Chile (en la sierra fronteriza con Bolivia como en las ciudades de Arica, Iquique y Antofagasta), y la parte norte de Argentina (en la sierra Jujeña y Salteña, y en las provincias del mismo nombre), además de la población migrante en la ciudad de Buenos Aires.

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 ultimo censo (INE 2012) de los 10.027.254 habitantes se denomina aymaras 1.191.352 y conforman 11.88 %.

El territorio donde se habla aymara estaba en plena expansión desde las migraciones masivas desde los años 50. Actualmente los aymaras hablantes, al igual que los quechuas, habitan en casi todo el territorio del Estado Plurinacional de Bolivia, con la globalización han emigrado a países vecinos como Argentina, Brasil España y Estado Unidos. Como resultado, se hallan expresiones culturales en esta regiones y en otras.

Si bien la región aymara está dividida políticamente por la fronteras nacionales entre Perú, Bolivia, Chile y Argentina, la comunidad de hablantes tienen mucho en común en su lengua como en su cultura, y las diferencias dialectales son mínimas como las diferencias culturales. En su gran mayoría, los hablantes aymaras en los países andinos son bilingües en castellano y algunos son trilingües con la adición del quechua y en algunos casos del uru. Sólo algunas personas mayores son aún monolingües en esta lengua.

El aymara es un idioma sufijante del tipo que a menudo se llama polisintético.

Los recursos gramaticales que están a disposición de los hablantes del idioma por medio de la sufijación son amplios y extensivamente utilizados; la cultura da gran valor al uso diestro de los recursos del idioma.

Lo apropiado en diferentes circunstancias está cuidadosamente definido, dentro de los límites de una situación determinada y teniendo presentes siempre las normas de respeto y cortesía, el hablar con elegancia, con ingenio e inteligencia, construyendo bellas figuras de dicción y juegos de palabras, es sumamente apreciado.

En cambio, el hablar mal induce sanciones negativas, siendo atribuido a menudo a la mala voluntad, estupidez o arrogancia del hablante. El hablar mal consiste en el uso de oraciones cortas con un mínimo de formas morfológicas y sintácticas; el expresarse brusca y secamente, ignorando los postulados que se mencionan a continuación; y el no prestar la debida atención a lo que otros dicen. La sanción negativa con la cual el aymara hablante puede castigar a la persona que habla mal suele ser el silencio, aunque el silencio también puede ser señal de cortesía hacia la persona que habla.

2.4.1. Gramática de la lengua Aymara

El alfabeto Fonémico del idioma aymara, está compuesto por 30 fonemas segméntales: 26 consonantes, 3 vocales (*/a/,/i/,/u/*) 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 estatus fonémico y como tal sirve para distinguir fonemas tanto radicales como sufijos; también verbaliza a nominales y además tiene el estatuto para lingüístico.

2.4.2. Símbolos Fonéticos

Existen varios símbolos con relación a los sonidos producidos por un hablante aymara y se detallan según el alfabeto fonético internacional.

Consonante	Bilabial	Alveolar	Palatal	Velar	Uvular	Glotal
Oclusiva simple	/p/	/t/		/k/	/q/	/'/
Oclusiva aspirada	/ph/	/th/		/kh/	/qh/	
Oclusiva glotalizada	/p'/	/t'/		/k'/	/q'/	
Africada simple			/ch/			
Africada aspirada			/chh/			
Africada glotalizada			/ch'/			
Fricativa		/s/			/χ/	/h/
Nasal	/m/	/n/	/ñ/			
Liquida lateral		/I/	/λ/			
Liquida vibrante		/r/				
Semi - consonante	/w/		/y/			

Tabla 2.2: Consonantes de Aymara [1, página 16]

A continuación se presenta el cuadro fonológico del aymara organizado por el modo y punto de articulación.

Vocal	Anterior	Central	Posterior
Alta	/i/		/u/
Baja		/a/	

Tabla 2.3: Vocales del Aymara [1, página 16]

En aymara no existe mayor problema en la escritura como en la pronunciación de las palabras: se lee casi tal como se escribe, excepto en los post velares [/q/, /qh/, /q'/, /x/] donde las vocales cerradas /i/ y /u se pronuncian como vocales abiertas [e, o]; esto ocurre por el condicionamiento de las consonantes post velares y no porque las vocales sean abiertas.

- Las vocales cerradas /i/ y /u/ antes o después de los fonemas post velares /q/, /qh/, /q'/ y /x/ se pronuncian como [e] y [o], y en algunos casos incluso mucho más abierto.

En aymara no existen diptongos ni triptongos como ocurre en castellano o en otras

lenguas, es decir, no existe la sucesión de las vocales; esto implica que no puede suceder una vocal a otra vocal diferente.

2.4.3. Los principales elementos gramaticales

Los principales elementos gramaticales de Pronombre Interrogativo, Sufijos Oracionales, singular y plural se detalla posterior:

Pronombre Interrogativo		Sufijos Oracionales	
Kuna	¿Qué?	-xa	Sufijo enfatizados
Khiti	¿Quien?	-wa	Sufijo afirmativo
Kawki	¿Donde?	-sa	Sufijo de información
Kamisa	¿Cómo?	-ti	Sufijo interrogativo “?”
Qawqha	¿Cuánto?	-sti	Sufijo de interrogación “?”
Kunaqata	¿Cuando?		
Kunapacha	¿Cuando?		

Sujeto	Singular	Sujeto	Plural
Yo	Naya	Nosotros(as)/Excluyente	Nanaka
Tu, usted, vos	Juma	Ustedes	Jumanaka
El, ella	Jupa	Ellos(as)	Jupanaka
Nosotros(as)	Jiwaxa	Todos nosotros(as)/incluyente	Jiwasanaka

Pronombre Demostrativos		Sufijos Posesivos	
Aka	Esto, esta, este	-ja	mi Posesivo 1ra Persona
Uka	Eso, esa, ese	-ma	tu Posesivo 2ra Persona
Khaya	Aquel, aquello/a	-pa	su Posesivo 3ra Persona
Khuri	Más, allá	-sa	mi nuestro/a Posesivo 4ta Persona

Sufijos de caso

-ru	Sufijo direccional	“a, hacia”
-ta	Sufijo de procedencia	“de, desde”

-taki	Sufijo destinativo	“para”
-mpi	Sufijo instrumentativo	“con”
-na	Sufijo causativo	“en”
-layku	Sufijo causativo	“por o a causa de”
-kama	Sufijo limitativo	“a, hacia”
-nki	Sufijo genitivo	“de”

Otros sufijos mas usuales

-raki	Sufijo aditivo	“también”
-puni/pini	Sufijo definitivo	“siempre”
-ki	Sufijo afectivo	“nomás”
-ra	Sufijo continuativo	“todavia”
-na	Sufijo causativo	“en”
-pi	Sufijo reafirmativo	“siempre”
-si	Sufijo reflexivo	“me, te, se, nos”
-tá	Sufijo de cortesia	“por fabor un momento”
-ya	Sufijo de cortesía	“por favor”
-ni	Sufijo de traslocativo	“ir a”
-naka	Sufijo pluralizador	“s”

Sufijos flexivos verbales

	Presente/pasado	Futuro
Primera persona	-tha	-¨
Segunda persona	-ta	-¨ta
Tercera persona	-i	-ni
Cuarta persona	-tan	-ñani

2.4.4. Símbolos ortográficos

Se refiere a los segmentos gráficos que se utilizan para escribir en la lengua aymara. Un segmento es un grafema o mas comúnmente llamado letra, cada una de ellas tiene

el rol de representar a una forma de escritura ortográfica. Algunos fonemas son a su vez simbolizados por dígrafos o trigrafos, otros son modificados o suprimidos de acuerdo a la convención social, tal como se describe en la siguiente tabla.

Consonante	Bilabial	Alveolar	Palatal	Velar	Uvular
simple	<p>	<t>	<ch>	<k>	<q>
Aspirada	<ph>	<th>	<chh>	<kh>	<qh>
Glotalizada	<p'>	<t'>	<ch'>	<k'>	<q'>
Fricativa		<s>		<j>	<x>
Nasal	<m>	<n>	<ñ>		
Liquida lateral		<l>	<ll>		
Liquida vibrante		<r>			
Semi consonante	<w>				

Tabla 2.4: Grafias Consonánticas [1, página 18]

Vocal	Anterior	Central	Posterior
Alta	<i>		<u>
Baja		<a>	

Tabla 2.5: Grafías vocálicas [1, página 18]

El alargamiento vocálico diéresis.

El diacrítico reductor.

En tal contexto, la ortografía aymara cuenta con un cuadro alfabético de 26 consonantes, 3 vocales, 1 alargamiento vocálico y 1 diacrítico reductor. En total son 31 grafías de escritura aymara.

En resumen, al observar todos los cuadros de arriba se distingue fácilmente el valor del símbolo, ya sea fonético, fonémico u ortográfico, a través de los corchetes [ ], barras / / o corchetes angulados <>respectivamente. Pues no se debe confundir, el nivel fonémico u ortográfico. Así mismo los símbolos fonémicos derivan del alfabeto fonético internacional y arriban hasta el alfabeto ortográfico del idioma aymara. Así el alfabeto aymara no es totalmente arbitrario, sino producto de estudios fonéticos

y fonémicos. Además, el aymara cuentan con la grafía <j>que proviene del fonema /h/ y ella forma parte de la escritura ortográfica del aymara.

2.4.5. Organización y escritura de sufijos

El idioma aymara también enfatiza la escritura morfémica porque esta lengua es tipo lógicamente aglutinante y morfológicamente relevante. Es así porque la yuxtaposición de los sufijos ocupa un lugar importante en la escritura de los enunciados u oraciones en este idioma. Por ende, veamos los cuadro de los sufijos respectivos.

Derivativo	**Fuerza morfemica**	**Sufijo**	**Ejemplo**
Nominal denominativo	Normal	-ni (posesivo)	Utani 'que tiene casa'
		-lla (deminutivo)	Utalla 'casita'
		-chi (diminutivo fútil)	Paquchi 'rubiecito'
		-chu (deminutivo familiar)	Waynuchu 'jovencito'
		-ku (deminutivo)	Tataku 'papito'
		-la (afectivo)	Qulila 'niña'
		-li (senectivo)	Awkili 'Ancianito'
		-lu (habitual)	Chíqatuu 'zurdo'
		-su (estimativo)	Tatasu 'papito'
		-wja (aproximativo)	Utawja 'cerca de la casa'
		-xa (localizador)	Akäx Ukäxa 'este y este lado'
	Profuerte	-pacha (inclusivo)	Markpacha 'todo el pueblo'
		-ch'a (dimensional)	Ukchá 'de este tamaño'
		-ch'i (parentesco)	Awkchí 'suegro'
		-sa (localizador)	Uks Uksa 'ese y ese lado'

	Semifuerte	-kati (cercativo)	Khaykati ‘enfrente’
		-kipa (alternativo)	Maykipa ‘Uno si, uno no’
		-wisa (privativo)	Nayrawisa ‘ciego’
		-rara (ponderativo)	Lakarara ‘hablador’
		-kucha (definitivo)	Jichhakucha ‘ahora si’
		-qata (ordinal)	Nayraqata ‘primero’
		-wiri (ordinal colectivo)	Nayrawiri ‘el primer grupo’
		-pacha (identitario)	Jiwaspacha ‘nosotros mismos’
Nominal deverbativo	Normal	-ña (infinitivo/nombre)	Pichaña ‘barrer/escoba’
		-wi (nombre abstracto)	Qalltawi ‘introducción’
		-ta (participio)	Munata ‘querido’
		-chi (extremativo)	Munachi ‘ambicioso’
	Profuerte	-kaya/kaña (extremativo)	Munkaya ‘posible de amar’
	Contrafuerte	-iri (agentivo)	Jaylliri ‘cantante’

Tabla 2.6: Sufijos derivativos o lexicos nominales [1, página 19]

Como se observa los nominalizadores de verbativos arriba, el sufijo -ña indica la forma de infinitivo de los verbos, pichaña ‘barrer’, qalltaña ‘empezar’, munaña ‘querer’, etc. A su vez -ña marca el nombre concreto o abstracto dependiendo de la semántica de dicha palabra, como ser pichaña ‘escoba’, chhaxraña ‘peineta’, son nom-

bres concretos, mientras que munaña, munasiña 'amor', jiwaña 'muerte'son nombres abstractos; además jakaña 'vida o placenta'puede ser abstracto o concreto, dependiendo de su significado según el uso contextualizado. El sufijo -wi opta normalmente por una forma oral [a:wi], [i:wi] o [u:wi] al ligarse a una vocal temática de la raíz verbal, siendo esto solo de carácter paralingüistico por la bemolización del sonido [w], pero que no varia de ninguna manera en el significado de la palabra nominalizada. Así al respecto ejemplificamos, sarawi 'costumbre', jakawi 'vida', sarnaqawi 'cultura', ikiwi 'sueño/ el dormí', jakhuwi 'numeración', entre otros.[1, página 11-25]

Derivativo	**Fuerza morfemica**	**Sufijo**	**Ejemplo**
Verbal derivativo	Normal	-nta (inductivo)	Ichuntaña 'llevar a un niño adentro'
		-qa (descensor)	Apaqaña 'bajar algun objeto'
		-qa (separativo)	Jithiqaña 'aislarse'
		-ni (translocativo)	Apaniña 'traer'
		-ni (translocativo distanciador)	Sataniña 'ir a sembrar'
		-li (oscilativo)	Aytiliña 'tambalear'
		-nki (oscilativo)	Aytinkiña 'tambalear'
		-ya (causativo)	Apayaña 'enviar'
		-si (reflexivo)	Sanjasiña 'peinarse'
		-si (reciproco)	Nuwasiña 'pelearse'
		-rapi (benefactivo)	Qullarapiña 'curaselo'
		-raqa (detrimentativo)	Jaytaraqaña 'dejárselo'

		-pa/pi (iterativo)	Isapaña 'Escuchar con frecuencia, entender'; laphapiña 'flamear'
		-ncha (desiderativo)	Saränchaña 'tener ganas de ir'
		-qi (hiperactivo)	Ch'isiqiña 'chillar demasiado'
		-waya (mocional)	Phuqhawayaña 'cumplir de paso'
		-yasi (asistivo)	Satayasiña 'ayudar a sembrar'
	Normal 1 / Profuerte 2	-1-ta/2-ta (mocional desergativo)	Irpatiña 'ir en pareja'; irpaqtaña 'bajar una pareja'
	Semifuerte	-kipa (perifactivo)	Imakipaña 'tapar por los bordes o por encima'
		-ra (contrafactivo)	Awkch'i 'suegro'
		-ra (serial)	Chinuraña 'amarrar uno en uno'
		-ra (estativo)	Sayaraña 'permanecer estable el agua'
		-paya (frecuentativo)	Muyupayaña 'andar alrededor de algo o alguien'
		-muku (distanciador)	Apamukuña 'abandonar algún objeto'
		-nuqa (expansivo)	Jalanuqaña 'distribuir'

		-rpaya (multiplicador)	Liwirpayaña 'derribar de golpe'
		-tata (propagativo)	Apatataña 'expandir, extender'
	Profuerte	-su (eductivo)	Jaqsuña 'arrojar afuera'
		-ta (ascensor discontinuo)	Aptaña 'levantar por un momento'
		-ta (incoactivo discontinuo)	Nukhtaña 'emoujar desde el inicio'
		-naqa (oscilativo)	Apnaqaña 'llevar de aquí para allá'
		-t'a (conativo)	Jiyt'aña 'jalar con fuerza por un momento'
		-t'a (atenuador)	Arst'am 'habla nomás tú'
		-thapi (congregativo)	Apthapiña 'recoger'
		-xasi (sostenedor)	Apxasiña 'sostener un objeto en la mano'
		-ch'uki (fijativo)	Unch'ukiña 'mirar fijando la vista'
		-xaru (incoactivo continuo)	Nukhxaruña 'empujar desde el inicio'
		-kata (atravesador)	Makataña 'pasar al frente'
		-kata (yuxtaposición/incompletivo)	Apkataña 'poner un objeto al lado de otro'

Verbal denominativo	Normal	-kata (ascensor continuo)	Apkataña 'elevar un objeto y fijarlo'
		-xata (superposición)	Apxataña 'poner un objeto encima de otro'
		-xata (completivo)	Jikxataña 'encontrar, alcanzar'
		-xaya (comitativo)	Parlxayaña 'hablar solo por compañía'
		-ja (partitivo/divisor)	aljaña 'vender'
		-nuqa (ubicador)	Apnuqaña 'colocar en algún sitio'
		-cha (factivo)	Pampachaña 'aplanar'
		-ncha (saturador)	Wayk'anchaña 'poner aji'
		-pta (transformativo)	Waliptaña 'sanarse'
		-nka (existencia móvil)	Utankaña 'estar en la casa'
		-¨ (existenicial fijo)	Utäña 'ser una familia'
	Semifuerte	-kipta (transformativo acelerado)	Tawaqkiptaña 'convertirse la niña en joven'
	Profuerte	-t'a (transformativo tardío)	Waytáña 'estar aniñado'

Tabla 2.7: Sufijos derivativos o léxicos verbales [1, página 21]

2.5. Metodologías

Los correctores ortográficos se han vuelto en una herramienta sumamente importante para los editores de texto, así como para otras aplicaciones, además que también se pueden utilizar para corregir la gran cantidad de información que actualmente esta en texto y debe ser capaz de soportar múltiples idiomas. Ante ese problema y por el tipo de datos que actualmente existen es que aparecen los espacios métricos que sumadamente con una función de distancia forman los accesos métricos. Para realizar el corrector ortográfico se ha elegido el Slim- Tree una estructura dinámica que tiene una serie de ventajas frente a otros accesos métricos y la distancia Levenshtein (Edit) que calcula el mínimo número de letras que hay que agregar, quitar o cambiar en una palabra para convertirla en otra.

2.5.1. Introducción

Las nuevas aplicaciones, como las bases de datos multimedias, biología computacional, recuperación de texto, reconocimiento de patrones, aproximación de funciones y compresión de audio video son cada vez más complejas y exigen distintas técnicas de indexación y búsqueda eficientes y no costosas en términos computacionales para cada caso, ante ese contexto aparecen los métodos de acceso métrico. El alcance de este artículo es de explicar un caso de estudio que es el problema del corrector ortográfico que es recuperar las palabras más próximas sin considerar la posición de la letra distinta, sea una letra de mas o de menos. Eso es lo que obliga pensar en una función de distancia y eso nos lleva a los espacios métricos.

2.5.2. Metodos de Acceso métrico

La evolución de la información y de las tecnologías de comunicación ha producido que repositorios de información se creen des estructuradamente, en donde se encuentran los textos libres. Ante eso surgen nuevas aplicaciones como minería de datos que requieren que el acceso a las bases de datos sea por cualquier campo y no sólo por las “llaves”, ante eso surgen los nuevos modelos de búsqueda por similitud o búsqueda por proximidad que son necesarios para resolver ese tipo de problemas. La búsqueda por similitud es modelada por una función de distancia que cumpla

con la desigualdad triangular. El espacio métrico de define como una colección de objetos asociados a una función de distancia que cumpla las siguientes propiedades: Sea a,b,c objetos de un espacio métrico.

- Simetría : $d(a,b) = d(b,a)$
- No negatividad : $d(a,b) > 0 si (a != b) y d(a,b) = 0 si (a = b)$
- Desigualdad triangular : $d(a,b) <= d(a,c) + d(c,b)$

El principal problema de los accesos métricos es la creación de una función de distancia, la cual es la más costosa de evaluar. El tiempo total para evaluar una consulta se divide de la siguiente forma:

$$T = número\ de\ evaluaciones\ de\ distancia\ *\ complejidad\ de\ d() +$$

$$tiempo\ extra\ del\ CPU\ +\ accesos\ al\ disco$$

El objetivo es minimizar T, buscando que la función $d()$ sea eficiente en términos computacionales, sin embargo en algunas aplicaciones la evaluación de $d()$ es tan costoso que los demás factores no se toman en cuenta. Es debido ha eso que los accesos a disco pasan a un segundo lugar en el momento de evaluar un acceso métrico, además este factor depende bastante de la cantidad de memoria principal que se tenga.

2.5.3. Estructuras de Indexación y Búsqueda

2.5.3.1. Tipos de Consultas

- Range Query $(q,r)d$: Consulta por rango. Recupera todos los elementos que están entre una distancia de r a q.
- Nearest neighbor Query NN(q) : Consulta a los vecinos más próximos. Recupera todos los elementos más cercanos a q. También se puede dar una distancia máxima r* tal que si el elemento más cercano se encuentra a una distancia mayor que r* no se reporta ningún elemento.
- k - Nearest neighbor Query NNk(q) : Consulta con los k vecinos más cercanos.

2.5.3.2. Árboles para Funciones Discretas

Hay dos tipos de distancias, las discretas y las continuas, para cada una de ellas hay estructuras de datos distintas. Cada una de ellas tiene su propio potencial al momento de trabajar con espacios métricos, a continuación se va ha presentar algunos accesos métricos para valores discretos y más adelante para valores continuos.

- BKT : El árbol Burkhard-Keller es probable el primer método de búsqueda es espacios métricos. Un pivote p es seleccionado de los datos y el conjuntos de datos se particiona en subconjuntos recursivamente.

- FQT : El árbol Fixed Queries es una variación del árbol BKT. La diferen- cia es que todos los pivotes están almacenados al mismo nivel, la ventaja es la forma de búsqueda en los nodos internos y los elementos se encuentran en la hojas.

- FHQT : Fixed Heigh. En este árbol todas la hojas están a la misma profundidad h, esto hace que algunas hojas sean más profundas de lo necesario debido a la comparación entre la consulta y el pivote en el nivel intermedio, sin embargo no hay costo adicional.

- FQA : Fixed Queries Array. No es un árbol sino una representación compacta del FHQT. Se dividen la hojas de la izquierda y derecha para poner los elementos en un arreglo. Por cada elemento del arreglo se calcula las distancias con los h pivotes, cada uno de estos h números se codifica en b bits y son concatenados en un número long, para que los niveles más altos del árbol sean dígitos más significantes.

2.5.3.3. Árboles para Funciones de Distancia Continua

- VPT : Vantage-Point Trees. Básicamente es un árbol binario recursivo en donde los pivotes se llaman puntos ventajosos, se particiona el espacio y los datos en esferas cortadas en cada nivel para proporcionar un filtrado de consultas similares.

- MVPT : Multiple Vantage Point. Es una generalización del VPT en un árbol multicaminos en donde se pueden elegir múltiples puntos ventajosos en cada nodo interno y cada uno se puede particionar los datos en m grupos.

- VPF : Otra generalización del VPT, Shorthand for Excluded Middle Van- tage Point Forest. El método consiste en excluir en cada nivel los elementos que se encuentran a una distancia intermedia a su respectivo pivote, es así que un segundo árbol es construido con la parte de la mitad del primer árbol excluido y así sucesivamente para obtener un bosque.

- BST : Bisector Tree. Es un árbol binario construido recursivamente. Dos objetos se eligen p1 y p2, en la inserción elementos cerca de p1 son inser- tados en la parte izquierda del subárbol y los cerca a p2 son insertados en el subárbol derecho. También se almacenan la distancia del elemento con sus padres, esto ayuda a la reducción de cálculos de distancias.

- GHT : Generalizad - Hyperplane Tree, es idéntico en construcción al BST, sin embargo su algoritmo usa hyperplanos entre dos pivotes p1 y p2 como criterio de división para las búsquedas.

- MT : Es una estructura con capacidades dinámicas, tiene un buen desem- peño en accesos a disco y requiere pocos cálculos de distancia. En este árbol se elige un grupo de elementos representativos que se sacan de ca- da nodo, los objetos representativos son organizados en un subárbol para poder guiar la inserción de un nuevo elemento. Cada elemento representa- tivo almacena su cobertura de radio para que al momento de la consulta se compare contra todos los representativos y si es que no se logra encontrar un elemento que represente donde pueda ir el nuevo objeto, el algoritmo recursivamente entra en donde no se ha podido descartar por el criterio de la cobertura del radio.

- Slim-Tree : Es un método de acceso métrico, que almacena los datos en las hojas, crece de abajo hacia arriba, los objetos son agrupados en páginas de tamaño fijo de disco, cada página corresponde a un nodo del árbol. Se diferencia de los demás métodos por el Minimal Spanning tree (MST), este algoritmo tiene mejor desempeño que las otras técnicas de partición, además existe un método

para guiar la inserción de un objeto a un nodo interno en un apropiado subárbol el cual aprovecha más la utilización de almacenaje, otro aspecto de innovación es el Slim Down que sirve para hacer más delgado y rápido al árbol, esto se aplica porque se sabe que una mayor sobre posición en una árbol métrico es responsable de la ineficiencia, para poder ajustar al árbol hay que medir la sobre posición usando el método del Fat-Factor y Relative Fat-Factor. Estas técnicas hacen que el Slim- Tree tenga mejor desempeño en las consultas.

2.5.4. Construcción del Slim- Tree

Se ha elegido el Slim- Tree por tener una serie de innovaciones y ventajas frente a otros accesos métricos, como el MST, Fat- Factor para calcular el nivel de sobre posición y el Slim- Down para adelgazar el árbol, es por esa razones que se va explicar la forma de construcción de este método. Se empieza desde el nodo raíz, el algoritmo trata de ubicar un nodo que pueda cubrir el nuevo objeto, si ninguno califica, se selecciona el nodo del cual su centro en el más cercano al nuevo objeto. Si más de un nodo califica se ejecuta el algoritmo ChooseSubTree para seleccionar uno de ellos, este proceso es recursivo y se aplica a todas las hojas del árbol. Cuando un nodo m esta sobre cargado, un nuevo nodo m' es insertado al mismo nivel, y los objetos son distribuidos entre los nodos, cuando un nodo incluyendo la raíz se parte se elige un algoritmo de partición.

2.5.4.1. Métodos del ChooseSubTree (Elegir Sub Árbol)

- random : Aleatoriamente se elige uno de los nodos calificados.

- mindlist : Escoge el nodo que tiene la menor distancia desde el nuevo objeto y el centro representativo del nodo.

- minoccup : Elige el nodo que minimiza la ocupación entre los nodos calificados, este es el método por defecto debido a su buen desempeño comprobado en el árbol MT.

2.5.4.2. Algoritmos de Partición del Slim- Tree

- random : Los dos nuevos objetos del centro son elegidos aleatoriamente y los objetos existentes son distribuidos entre los demás. Cada objeto es almacenado en un nuevo nodo que tiene su centro más cercano ha ese objeto. Es una estrategia bastante rápida.

- minMax : Todos los posibles pares de objetos son considerados como representativos potenciales, para cada par existe un algoritmo lineal que asigna a los objetos un representativo. El par que minimiza la cobertura del radios es elegido. Este algoritmo ya ha sido usado en el árbol MT y se ha encontrado que es el algoritmo más adecuado.

- MST (Minimal Spanning Tree) : Se genera la dispersión de los objetos o el grafo y el arco más largo generado se saca, este método produce que el Slim-Tree sea igual de bueno que el minMax en fracciones de tiempo.

2.5.5. Metodologia para Correctores Ortográficos

Los correctores ortográficos se han convertido en una herramienta esencial en los editores te texto y en otras aplicaciones, lo que permite la calidad de la redacción en los textos. Actualmente estos deben contar con la facilidad de corregir una palabra sin importar el idioma, aparentemente la solución es trivial pero surge el problema de los idiomas en donde las palabras no son parte de un alfabeto, sino son símbolos, este problema no se va ha explicar en este artículo, sino mas bien se va enfocar en la solución para idiomas que utilicen el alfabeto latino.

2.5.5.1. Método de prefijos o sufijos

Esta técnica consiste en usar los prefijos o sufijos que están en la raíz de las palabras para formar otras palabras. Por ejemplo la palabra supply en ingles, si se saca el último carácter y y se le agrega la cadena ied que es un sufijo, tenemos la palabra supplied. A continuación se muestra un ejemplo de cómo se define los sufijos en ingles que utiliza el corrector ortográfico del OpenOffice org. En los cuatro campos:

- SFX indica que es un sufijo.
- D es el nombre del carácter que representa el sufijo.
- Y indica que se puede combinar con prefijos támbien.
- 4 especifica que son 4 secuencias o formas que existen para esos sufijos y prefijos para formar otras palabras.

En el ejemplo (y) indica el carácter que se debe sacar antes de agregar el sufijo (0 indica la cadena vacía), luego ied es la cadena de sufijos que se debe agregar y aeiou son las condiciones que se debe verificar antes de aplicar el sufijo en la Figura 2.13.

```
SFX D Y 4
SFX D   0     d          e
SFX D   y     ied        [^aeiou]y
SFX D   0     ed         [^ey]
SFX D   0     ed         [aeiou]y
```

Figura 2.13: Corrector Ortografico. [5, página 6]

Ejemplo de una declaración de sufijos que utiliza el OpenOffice La ventaja de este método es que esta comprobado que funciona, pero su desventaja es que requiere un archivo con el diccionario y otro con el formato de los sufijos el cual es bastante tedioso formularlo.

2.5.5.2. Distancia Levenshtein

Conocida también como distancia Edit, devuelve la distancia que hay entre dos palabras, por ejemplo la distancia entre lima y limonada es de , es decir es el mínimo número de letras que hay que agregar, quitar o cambiar en una palabra para convertirla en otra. Esta distancia computacionalmente es cara debido a que la complejidad del algoritmo es de O(mn) donde m, n son las longitudes de las palabras comparadas. Sin embargo la ventaja es que el tiempo de comparación usando un acceso métrico es bastante bueno y solo requiere un archivo diccionario para hacer las comparaciones.

2.5.6. Morfología de Estados Finitos

El procesador morfológico es una herramienta básica para el **Procesamiento de lenguaje Natural**. Si el idioma es de flexión rica una lista de palabras con su análisis correspondiente no es una buena solución.

La propuesta más adecuada para crear un procesador morfológico basado en la morfología de dos niveles para un idioma aglutinante de dos niveles para un idioma aglutinan-ate como el euskera era utilizar la morfología de dos niveles propuesta por Koskenniemi. El exitoso Xuxen esta basado en esta tecnología. La morfología se describe por medio de dos ficheros, el léxico, donde se describen por medio de dos ficheros, el léxico, donde se describen los morfemas y los conjuntos de morfemas y los conjuntos de morfemas que les puede seguir (morfotáctica); y las reglas fonológicas, que describen los cambios producidos al encadenar los morfemas.

La ideas de la morfología de dos niveles fue tomada y mejorada por Xerox dando lugar a la morfologia de estados finitos (Beesley y Karttunenn, 2003). La diferencia fundamental entre la propuesta de Koskenniemi y la posterior evolución en Xerox es que permiten tanto reglas paralelas como reglas secuenciales para expresar las reglas fonológicas.

Las reglas paralelas tienen la ventaja de que el orden no es significativo y que no hace falta definir lenguajes intermedios entre las palabras del texto (nivel superficial) y la representación léxica (nivel léxico). Sin embargo las reglas paralelas deben tener en cuenta en sus contextos los efectos de las reglas relacionados, y esto muchas veces es fuente de errores. ambos tipos de reglas pueden ser compiladas y convertidas a transductores muy eficientes. Veamos un ejemplo en euskera. El prefijo *ber* antecede a la raíces verbales. delante de vocal la erre se dobla (es *r* fuerte) y adelante de consonante se transforma en *bir*. Además delante de una *h*, hace que la *h* desaparezca. Si representamos el prefijo como *beR* (marcando la r fuerte), tenemos que de *beR+hasi berrasi*. Usando reglas paralelas (simplificadas) tenemos la descripción del ejemplo siguiente:

```
R->r; +->0;
h->0 || b e R:r 0: r +:0 _ Vocal;
0->r || R:r _ +:0 (h:0) Vocal;
e->i || b _ R +:0 Cons;
```

Usando reglas secuenciales (simplificadas) tenemos la descripción del siguiente ejemplo.

```
h->0 || b e R + _ Vocal;
  0->r || R _ + Vocal;
 e->i || b _ R + Cons;
```

`R->r ; +->0 ;` El segundo grupo de reglas son más sencillas, pero su orden es significativo. EL toolkit de Xerox proporciona herramientas para reglas paralelas (*twolc*) y reglas secuenciales (*xfst*), además de para el léxico (*lexc*). Su tendencia que se ha visto confirmada en posteriores implementaciones.

2.5.7. Morfologia en Software Libre

Las herramientas Xerox son de gran calidad y consiguen transductores muy compactos, pero tienen una gran inconveniente; la licencia es muy restrictiva y salvo excepciones no puede ser usada para aplicaciones comerciales. Ante ello se han desarrollado en software libre herramientas que pueden sustituir las de Xerox. Queremos destacar dos: *hunspell y foma*. *Hunspell* no trabaja con transductores, pero es una solución mejor que sus predecesores *ispell, aspell y myspell*,ya que permite mayor número de paradigmas y doble encadenamiento de sufijos. No acepta reglas paralelas ni secuenciales independientes del léxico, ya que los cambios deben ser explicitados en la que descripción del léxico. Su mayor interés es que esta aceptado por las ultimas versiones de OOffice y Mozilla, por lo que una descripción para un idioma usando *hunspell* deviene automaticamente en un corrector ortográfico para estas herramientas. *Foma* es un toolkit que quiere ser equivalente a las herramientas *xfst* y *lexc* de Xerox pero con una implementación independiente y licenciado bajo General Public License GPL (*http://foma.sourceforge.net*). Por lo tanto es adecuado para descripciones de morfologia de estados finitos usando reglas secuenciales.

Según su autor las descripciones para Xerox funcionan directamente, y la compilación es más eficiente. Tiene una ventaja adicional, y es que las descripciones preparadas para *lexc* y *xfst* de Xerox se compilan directamente en *foma* (salvo alguna excepción y siempre que los datos en Unicode); ya que esta nueva herramienta integra los mismos comandos y la misma sintaxis que las herramientas citadas.

2.5.8. El Procesamiento del Lenguaje Natural

El lenguaje es sin duda una pieza clave en el desarrollo de los seres humanos. En su versión escrita nos aporta un archivo de conocimiento que permite traspasar generaciones de manera que se aprovechen y disfruten los logros alcanzados por nuestros antepasados, mientras que en su faceta hablada, es el sustento de la interacción entre personas y nos permite coordinarnos en nuestro comportamiento diario.

Dada su trascendencia, existen varios colectivos que estudian el lenguaje desde diferentes perspectivas (lingüistas, psicolingüistas y filósofos por ejemplo), cada uno afrontando su tarea con sus propios objetivos. En el área del Procesamiento del Lenguaje Natural (PLN) se estudian técnicas que permitan crear modelos del lenguaje con el objetivo de que un sistema informático maneje el lenguaje natural en determinadas y muy diversas tareas. El informático por lo tanto persigue la creación de un modelo computacional que refleje con la mayor fidelidad y precisión posible las características del lenguaje.

Una primera división en el tipo de trabajos que se llevan a cabo a este respecto, puede hacerse entre las tareas que se encargan de reconocer las palabras escritas o habladas y las tareas que se llevan a cabo una vez reconocidos los elementos léxicos sobre los que trabajar. En el primer conjunto se encontraría por ejemplo el tratamiento de señales que trata de reconocer las palabras que pronuncia un interlocutor de los sonidos que emite al hablar, teniendo que tratar con diferentes características vocales, pronunciación, deficiencias y demás problemas derivados del habla. Nosotros nos centraremos en el segundo conjunto, por lo que daremos por reconocidas las palabras con las que trabajaremos, independizándonos de su vıa de entrada, que puede ser manuscrita, hablada o mecanografiada.

El interés en estudiar el lenguaje es generalmente un interés de doble vía, ya que por un lado está el analizar la naturaleza del lenguaje y estudiar cómo funcionan, y por otro lado, el crear sistemas capaces de interaccionar con el hombre. Esta interacción a su vez permitiría a los sistemas informáticos acceder a una cantidad inmensa de información escrita y disponible en lenguaje natural, mientras que a su

vez, permitiría que estos sistemas fueran accesibles a cualquiera, permitiendo la comunicación con ellos de una forma que no nos puede resultar más natural y cercana, como es empleando el lenguaje natural.

Sin embargo, para que este escenario pueda lograrse hay que superar muchos retos y afrontar los numerosos desafíos que presenta una técnica tan compleja como el lenguaje. El conocimiento relacionado con el entendimiento del lenguaje natural se diversifica en varias ramas que pueden verse como diferentes tipos de conocimiento relevantes para el lenguaje natural o como características del conocimiento asociado al lenguaje:

- Conocimiento fonético: relaciona las palabras y los sonidos que las producen.
- Conocimiento morfológico: trata la construcción de palabras a partir de elementos más básicos como son los morfemas.
- Conocimiento sintáctico: dicta como pueden unirse las palabras para formar frases y el papel que desempeñan dentro de la frase.
- Conocimiento semántico: estudia el significado de las palabras y el significado que adquieren la frases sin considerar el contexto.
- Conocimiento pragmático: se encarga de distinguir las diferentes situaciones en que puede usarse una frase y como afecta esto a su interpretación.
- Conocimiento de discurso: hace referencia a como una frase afecta a las frases que la rodean.
- Conocimiento de mundo: incluye todo lo necesario para que se pueda mantener una conversación (la estructura del mundo que nos rodea, las creencias y objetivos de nuestro interlocutor, etc).

En función de lo que estemos estudiando, deberemos considerar diferentes aspectos de todos los arriba mencionados y superar los problemas que cada unos nos pueda presentar. Para el estudio se escogerá el Conocimiento Morfologico.

2.5.9. Características morfológicas del Castellano

El castellano es una lengua que derivó del latín y tiene una gramática compleja. Para llevar a cabo la construcción de cualquier plataforma léxica la primera tarea que se debe llevar a cabo es el estudio de la gramática castellana. Este estudio debe permitir formalizar el conjunto mínimo de reglas necesario que permita extraer el conjunto de palabras reconocidas por la lengua castellana a partir de un conjunto de palabras raíces mínimo. Para alcanzar dicho objetivo se construyó un árbol de derivación a partir de una palabra raíz. Una versión simplificada de dicho árbol se muestra en la Figura 2.14.

Los principales problemas que se encontraron en la construcción de este conjunto de reglas de derivación vinieron originados por las características del castellano. Los mas relevantes se exponen a continuación:

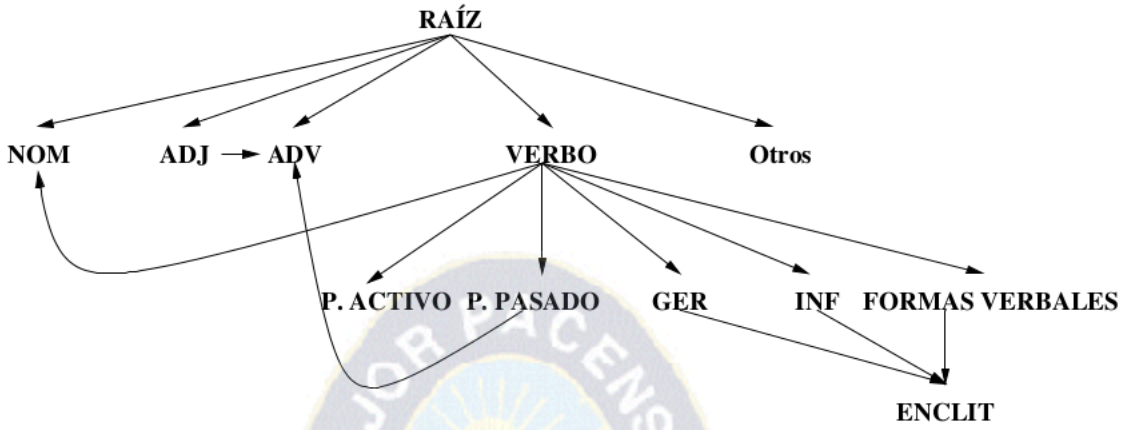

Figura 2.14: Estructura Morfológica simplificada del castellano. [8, página 3]

2.5.9.1. Derivaciones de género y número

Los adjetivos (ADJ) y sustantivos (NOM) tienen género (masculino o femenino) y número (singular y plural). La situación habitual es que un adjetivo o nombre tenga derivación tanto en género como en número. Por ejemplo: *perro* → *perra* y (*perros, perras*). Otros casos únicamente tienen un género y, por tanto sólo admiten derivación en número. Es el caso de un sustantivo masculino como *álamo, alamos*, o femenino como *casa, casas*.

2.5.9.2. Conjugación verbal

Cada una de las tres conjugaciones verbales del castellano tienen 40 derivaciones temporales (P. ACTIVO, P. PASADO,GER, INF y FORMAS VERBALES). Como es sabido, los verbos regulares tienen un conjunto estricto de reglas de derivación que son idénticas para todos los de una misma conjugación. Los verbos irregulares tienen al menos una derivación diferente que las derivaciones regulares correspondientes a su conjugación. Las derivaciones irregulares se agrupan en alrededor de 100 tipos diferentes de irregularidades.

2.5.9.3. Formas enclíticas

Algunas derivaciones verbales se generan añadiendo una forma pronominal al final de una forma verbal (ENCLIT). En el castellano escrito se pueden encontrar dos formas enclíticas diferentes: los verbos pronominales, cuyas formas enclíticas se generan añadiendo los sufijos -me, -te, -se, -nos y -os en el infinitivo y en el gerundio (amar → amarte), y los verbos transitivos, cuyas formas pronominales se generan añadiendo las terminaciones -lo, -la, -los, -las, -le y -les (amar → amarla). Ambas formas enclíticas se pueden combinar para formar enclíticos más complejos (ajustar → ajustármelo). Esto genera un conjunto de reglas de complejidad $O_{(n^2)}$. Además, estas formas enclíticas se ven afectadas por las irregularidades que presentan algunos verbos en su gerundio (vestir → vistiéndote), lo que incrementa el grado de complejidad para las formas enclíticas.

2.5.9.4. Nombres derivados de verbos

Algunos sustantivos son forma derivadas de un verbo *imaginar* → *imaginación* o *abatir rightarrow abatimiento* (VERBO → NOM).

2.5.9.5. Adverbios derivados de adjetivos

Gran parte de los adverbios modales se generan añadiendo el sufijo *-mente* a una adjetivo (ADJ → ADV): *tranquilo* → *tranquilamente*.

2.5.9.6. Superlativos y diminutivos

Las formas regulares de superlativos se forman añadiendo en sufijo *-ísimo* a un adejtivo (*grande* → *grandísimo*). Los diminutivos se forman añadiendo los sufijos *-ico, -ito* y *-illo* a un nombre adjetivo o nombre.

2.5.9.7. Vocales acentuadas

Hay mucahas particularidades relacionadas con las derivaciones de género y número que se han tenido en cuenta al hacer el estudio del modelo. algunas palabras pierden una vocal acentuada sustituyéndola por su equivalente no acentuada: *gañán, gañanes*. Teniendo en cuenta las características descritas en los párrafos anteriores

se ha desarrollado un conjunto de reglas formales que comprende un extenso subconjunto de las que conforman la gramática castellana. Cada una de las entradas del diccionario que contiene las palabras raíces tiene una etiqueta que representa una lista de reglas de derivación que se deben aplicar a dicha palabra para obtener sus formas derivadas.

2.5.10. Implementación del modelo

Las características gramaticales estudiadas en el apartado anterior han llevado a la realización del modelo ajustándose a las restricciones que imponía la herramienta ispell. Estas restricciones se basan en agrupar un conjunto de reglas(clase) al que se asocia una etiqueta que ser a referenciada en las etiquetas del diccionario raíz.

Puesto que el uso del castellano se basa en gran parte en las formas deriva- das (un verbo castellano tiene 55 formas derivadas), las reglas de derivación del diccionario incluyen todas las derivaciones de los verbos regulares. Además, se han incorporado reglas adicionales para tener en cuenta la mayor parte de los patrones por los que se rigen las derivaciones de los verbos irregulares. La inclusión de las formas derivadas de los verbos ser, ir, haber y estar se han incluido en su totalidad en el diccionario raíz al no adecuarse fáacilmente a ninguno de los patrones considerados. En resumen, el conjunto de reglas implantado para especificar la gramática castellana contiene alrededor de 3.300 reglas agrupadas en 57 macroreglas o clases. Cada macroregla que se describe a continuación refleja un aspecto particular de la gramática castellana que se ha descrito en la sección anterior. Cada una de las reglas de derivación que componen una clase o macroregla. La condición que se muestra en la primera columna de cada uno de los ejemplos representa la aceptación de una palabra para ejecutar la acción que se muestra en la segunda columna. Si una palabra termina con el sufijo especificado, se realiza la acción subsiguiente. Esta acción se basa en sustituir un morfema de la palabra raíz por otro, o simplemente añadir un morfema adicional.

2.5.10.1. Derivaciones de género y número

Se han incluido dos macroreglas que realizan estas derivaciones. La derivaciones en número incluyen 11 reglas.

La regla a aplicar depende de la terminación de la palabra raíz sobre la que se aplica la regla. La macroregla de derivación en género y número se compone de 20 reglas. A continuación se muestran algunos ejemplos de derivación de estas clases en la Figura 2.15:

Verbos Regulares			**Verbos Irregulares**		
Condición	Acción	Ejemplo	Condición	Acción	Ejemplo
AR	-AR, O	amo	IAR	-IAR, ÍO	envío
CER	-CER, ZO	venzo	OÑAR	-OÑAR, UEÑO	sueño
CIR	-CIR, ZO	zurzo	SABER	-ABER, É	sé

Figura 2.15: Verbos Regulares e Irregulares. [8, página 5]

Estas reglas no han tenido en cuenta los verbos *ser, estar, ir* y *haber* puesto que no existen un conjunto de patrones que permitan derivar todas sus formas verbales a partir del infinitivo. Todas sus formas derivadas se han incluido explícitamente en el diccionario de palabras raíces.

2.5.10.2. Formas enclíticas

Los verbos regulares incluyen alrededor de 200 reglas de derivación para generar las formas enclíticas, mientras que los verbos irregulares incorporan en torno a 400. Estas reglas representan los enclíticos generados por las derivaciones pronominales, transitivas y combinadas de ambas. Estas reglas únicamente se aplican a las formas del gerundio e infinitivo.

2.5.10.3. Nombres derivados de verbos

Se han tenido en cuenta los nombres acabados en *-miento* y *-cion* que se derivan de verbos a partir de dos macroreglas.

2.5.10.4. Adverbios derivativos de adjetivos

Adverbios derivados de adjetivos. Se ha considerado una macroregla (clase) que genera los adverbios terminados en -mente.

2.5.10.5. Superlativos y diminutivos

Actualmente únicamente los superlativos regulares se han considerado y constituyen una clase.

2.5.11. Generación del Diccionario

El léxico para esta plataforma ha sido extraído de un Corpus de Españnol compilado por los autores. Este corpus, que contiene más de 20 millones de palabras, incluye textos extraídos de las siguientes fuentes: Textos de perióodicos españoles (ABC Cultural, El Mundo, El Periódico, etc.); Libros seleccionados, como la Biblia; Textos técnicos (informes técnicos, artículos, proyectos de fin de carrera, diccionarios técnicos y libros); Corpus oral; Versión concisa del diccionario Español-Inglés Collins.

El léxico básico resultante contiene más de 80.000 palabras distintas, 53.000 de las cuales han sido ya etiquetadas de acuerdo a las reglas de derivación mostradas en la sección anterior. Las restantes 27.000 están en proceso de etiquetado. El etiquetado se hace de forma semiautomática mediante una herramienta que extrae los morfemas de cada palabra y propone una o varias etiquetas tentativas. Sin embargo, las formas derivadas son comprobadas manualmente exico de referencia usado para verificar la correccion o no del etiquetador. El léxico de referencia usado para desarrolladores COES es el diccionario de la Real Academia Española institución oficial que vela por la pureza del lenguaje español y admite las nuevas palabras del mismo.

La versión de libre distribución de COES incluye un fichero de afijos y varios ficheros de léxico:

- Español.words contiene una lista de palabras del diccionario oficial de Español.
- Español.comp contiene una lista de palabras que no aparecen en el diccionario oficial de la Lengua Española, pero de uso habitual en los textos técnicos.
- Antiguas.words contiene una lista de palabras que aparecen en el diccionario oficial de Español, pero etiquetadas como palabras en desuso.

- Eespañol.nofl contiene una lista de palabras que no aparecen en el diccionario oficial de Español, pero que han sido frecuentemente encontradas en el corpus.
- Español.propios contiene una lista de nombres propios.

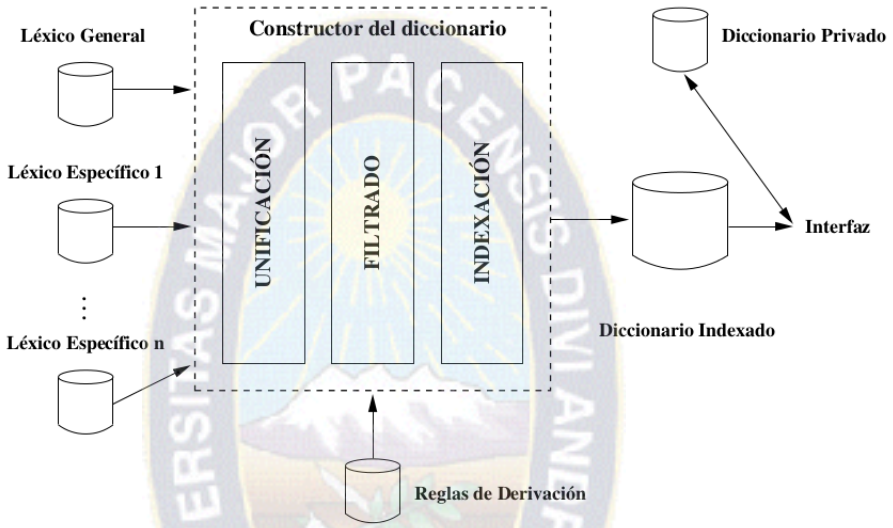

Figura 2.16: Generación del Diccionario. [8, página 7]

Cuando se aplican las reglas de derivación al léxico básico usado en COES actualmente se crea un diccionario que contiene más de 650.000 palabras. El diccionario de español se construye usando la herramienta ispell, que aplica las reglas de derivación elaboradas por los autores, siguiendo el formato de esta herramienta, al léxico básico. ispell sigue cuatro pasos básicos para generar del diccionario en la Figura 2.16:

- Generación de las reglas de derivación a partir del fichero de reglas.
- Unificación de las entradas del diccionario para evitar redundancias y formas ilegales.
- Interpretación de las reglas de derivación para calcular las formas derivadas.
- Construcción de un árbol indexado para conseguir una búsqueda eficiente en el diccionario.

El diccionario se puede generar en cualquier sistema operativo para el que exista una versión de ispell. Esto incluye cualquier computador que ejecute alguna versión de Unix y, ademas los sistemas Windows.

2.6. Distribución t de Student

2.6.1. Teoría de pequeñas muestras

En probabilidad y estadística, la **distribución t** o **distribución t de Student** es una distribución de probabilidad que surge del problema de estimar la media de una población normalmente distribuida cuando el tamaño de la muestra es pequeño.

A la teoría de pequeñas muestras también se le llama teoría exacta del muestreo, ya que también la podemos utilizar con muestras aleatorias de tamaño grande.

Veremos un nuevo concepto necesario para poder entender la distribución t Student. Este concepto es "grados de libertad".

Para definir grados de libertad se hará referencia a la varianza maestral:

$$S^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \tag{2.1}$$

Esta fórmula está basada en $n-1$ grados de libertad. Esta terminología resulta del hecho de que si bien s^2 está basada en n cantidades $x_1 - \bar{x}, x_2 - \bar{x}, ...x_n - \bar{x}$, éstas suman cero, así que especificar los valores de cualquier $n-1$ de las cantidades determina el valor restante. Por ejemplo, si $n = 4$ y $x_1 - \bar{x} = 8$; $x_2 - \bar{x} = 6$ y $x_4 - \bar{x} = 4$, entonces automáticamente tenemos $x_2 - \bar{x} = 2$, así que sólo tres de las cuatro medidas de $x_i - \bar{x}$ están libremente determinadas, la otra debe tomar el valor que haga esta suma cero; es por esto que solo tenemos 3 grados de libertad.

$$gradosdelibertad = n\acute{u}merodemediciones - 1$$

$$n_1 \; y \; n_2$$

$$x_1 = \frac{\sum_{i=1}^{n_1}(x_i)}{n_1}$$

$$x_1 = \frac{\sum_{i=1}^{n_2}(x_i)}{n_2}$$

- Dando un α

$$\alpha = 0{,}05$$

- Region de rechazo $1 - \alpha = 0,95 \rightarrow t_{(n_1+n_2-2,0,95)}$,
- Puesto que n es pequeño usaremos la distribución $\mathbf{t}$. $S^2 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}$
- Cálculo de S.

$$S_1^2 = \frac{\sum_{i=1}^{5}(x_i - \bar{x})^2}{n-1}$$

$$S_2^2 = \frac{\sum_{i=1}^{5}(x_i - \bar{x})^2}{n-1}$$

$$t_c = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}(\frac{1}{n_1}+\frac{1}{n_2})}}$$

- Conclusión ya que la t_c no esta en la región critica no se rechaza H_0, si esta en la región se recha H_0.

Capítulo 3

DISEÑO METODOLÓGICO

3.1. Introducción

La corrección es difícil, requiere de un profundo conocimiento de palabras y clasificarlas en sustantivos, adjetivos, verbos, adverbios y palabras especiales, para esto se usa la morfologia de estados finitos para representar en un Lexicon y dividirlas las palabras.

Como también la composición y la estructura de una palabra, para esto se usa las caracteristicas morfológicas del idioma aymara, agregando sufijos a la palabra raiz.

Para el desarrollo se usara *FOMA* en base a morfologia de estados finitos.

El problema es que idioma Aymara existen las elisiones vocálicas en el cual cumpliendo reglas de la gramática Aymara se va reemplazando, para esto existen reglas que eliden las vocales o consonates con *FOMA*.

La computadora o maquina debe leer la palabra original, comprobar si esta en el diccionario de datos (el cual estara contenida en un archivo lex todo un diccionario de aymara, es decir las palabras raiz), detectar si es una palabra conocida y si tiene errores, y mostrar las diferentes posibilidades de la corrección de la palabra.

3.1.1. Planificación Lingüística

El lexc y xfst son herramientas que permiten al desarrollador afirmar hechos lingüísticos formales de las notaciones que ya son familiares para los lingüistas formales. Así lexc nos da un marco para la especificación de los hechos morphotacticos en términos de sublexicos, entradas y clases de continuación; y xfst reescribir reglas proporcionan una manera de especificar datos fonológicos o alteraciones ortográficas. Pero estas herramientas son sólo marcos para afirmar hechos; ellos no le pueden decir cuáles son los hechos. La disciplina que estudia las lenguas naturales, el descubrimiento de sus normas y la formalización de los mismos, es la lingüística formal.

3.1.1.1. Ligüística Formal

En el lenguaje común, un lingüista es el que habla varios idiomas, es decir, un políglota, que es algo muy diferente. La capacidad de hablar varios idiomas, aunque admirable, no hace que un lingüista formal, más que tener un corazón que hace un cardiólogo.

Una gran cantidad de buenos lingüística formal se hizo antes de las herramientas informáticas disponibles, y una gran cantidad de mala lingüística formal se puede hacer incluso con las mejores herramientas computacionales. Es importante darse cuenta de que la codificación en lexc y xfst no es la lingüística, la formalización de los malos modelos lingüísticos simplemente resulta en malos programas. La lingüística formal es lo que siempre ha sido, el estudio formal y la comprensión de las estructuras lingüísticas ; y que el estudio y la comprensión debe ser lo primero . Después de la lingüística formal se realiza , después de un modelo se elaboró, implementamos y evaluamos el modelo de uso de las herramientas de estado finito.

3.1.2. Método de prefijos o sufijos

Ya que el idioma Aymara es netamente sufijante se escogio este metodo por que la Morfologia de Estados Finitos del foma se adecua a este metodo. Para lo cual se emplea el Procesamiento de Lenguaje Natural y el Conocimiento Morfologico para

el estudio.

3.1.2.1. Clasificación morfológica del Aymara

El aymara es una lengua originaria que tiene una gramatica compleja. Esta parte permite formalizar el conjunto minimo de reglas de formación de una palabra raiz y los sufijos. En el siguiente figura se detalla Estructura Morfológica del Aymara en la Figura 3.1.

Figura 3.1: Estructura Morfológica del Aymara [Elaboración Propia]

3.1.2.2. Derivaciones número

Los adjetivos (ADJETIVOS) y nombres (SUSTANTIVOS) tienen número (singular y plural). La situación habitual es que un adjetivo tiene la derivación *jiská* → *jiskálla* o nombre tenga singular y plural. Por ejemplo: *anu* → *anunaka*.

3.1.2.3. Conjugación verbal

Cada una de las conjugaciones verbales del aymara tienen. Como es sabido, los verbos regulares tienen un conjunto estricto de reglas de derivación que son idénticas para todos los de una misma conjugación. Los verbos irregulares tienen al menos una derivación diferente que las derivaciones regulares correspondientes a su conjugación. Ejemplo del verbo, con su pasado, presente, futuro *anataña* → *anatanawa, anatiwa, anataniwa*.

3.1.2.4. Formas enclíticas

Algunas derivaciones verbales se generan añadiendo una forma pronominal al final de una forma verbal (ENCLIT). En el castellano escrito se pueden encontrar dos formas enclíticas diferentes: los verbos pronominales, cuyas formas enclíticas se generan añadiendo los sufijos -wa, -xa, -sa, -sti, cha y -ti.

3.1.2.5. Alargamiento Vocalica

Hay muchas particularidades relacionadas con el alargamiento vocalica. Ejemplo alargando *Mä qarwa*.

3.1.2.6. Elisión Vocalica

Hay muchas particularidades relacionadas con las elisiones vocalicas. Ejemplo elidiendo la ultima letra de una palabra *Jumaxa* → *Jumax*.

3.2. Componentes del Corrector

Se realiza una clasificación de componetes que son los siguientes:

Diccionario de Palabras, en el cual estará contenido en un archivo lex dividido en Sustantivos, Verbos, Adjetivos, Adverbios, Pronombres y Pronombres Interrogativos a esto se lo llama palabra raíz. El cual sera extraido del libro Diccionario Aymara Castellano - Castellano Aymara de Felix Layme, de la siguiente manera.

```
DECLARACION DE SIMBOLOS Y CARACTERES MULTIPLES
DECLARACION DE SECCION
LEXICON Root
LEXICON Sustantivos
LEXICON Verbos
LEXICON Adejetivos
LEXICON Adverbios
LEXICON Pronombres
```

```
LEXICON PronombresInterrogativos
```

Composición de Palabras con sufijos, en el cual esta contenido en un archivo lex con el cual se componen los sufijos en base a la palabra raíz y los sufijos que pueden componerse formando el otra palabra distinta.

```
achachi[Sustantivo]:achachi CC_sustantivos;
LEXICON CC_sustantivos
CC_sustantivos_principales;
CC00;
LEXICON CC_sustantivos_principales
CC_sustantivos_pri;
CC_sustantivos_seg;
CC_sustantivos_ter;
CC00;
LEXICON CC00
#;
```

Definiendo Elisión Vocálica, en el cual esta contenido en un archivo foma, en el que se declara con el comando define la elisión cada vez que encuentre th +/@ con wa se elida la letra a.

```
define Elision04 a ->0 || t h _%+ w a„
a ->0 || t h _%@ w a ;
```

Definiendo Alargamiento Vocálico, en el cual esta contenido en un archivo foma, que cada vez que encuentre iya o aya lo reemplace por ä.

```
define Variante05 {iya} | {aya} (->) {ä} ;
```

Definiendo Reglas de corrección y reglas de Teclado, en el cual esta contenido en un archivo foma, en el cual se detalla si v l o y lo reemplaza por ll.

```
define TecladoL l | y (<-) ll ;
```

La palabra raíz es la parte para conformar las otras palabras agregando los sufijos para estructurar la nueva palabra con distinta significación.

Los sufijos permiten adicionar significado y clasificar la palabra.

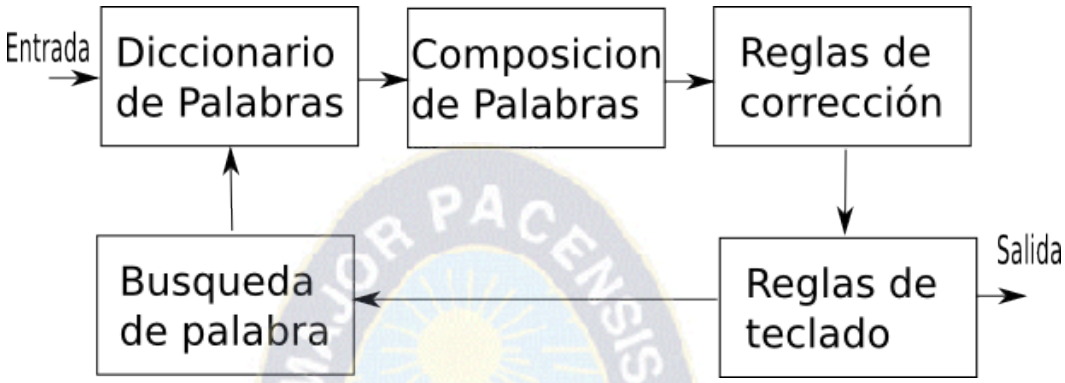

Figura 3.2: Esquema general del corrector ortográfico. [Elaboración Propia]

La estructura general del corrector Figura 3.2 nos permite corregir la palabra en el idioma aymara, por ello es muy importante formalizar todo el proceso, de manera que el proceso mismo sea mas entendible.

En la Figura 3.2 se formaliza todos los procesos mencionados tomando encuentra las entradas, el proceso mismo y la salida.

Después de realizar todos los anteriores pasos se proseguirá con la corrección de la palabra tomando encuentra las reglas del orden de sufijos, formalizados reglas lo que permite la corrección de la palabra en el idioma aymara.

Diccionario de Palabras: En este módulo se realizara un análisis de la palabra clasificándola en Sustantivos, Verbos, adjetivos, Adverbios, Pronombres y Pronombres Irregulares. Y para esto se tomo la obra de Felix Layme [4, Todo el libro].

Composición de Palabras: En este módulo se procederá a armar las palabras con los sufijos y sera distintas composiciones para Sustantivos, Verbos, Adjetivos, Adverbios, Pronombres y Pronombres Irregulares, en la Figura 3.3 se muetra una composición.

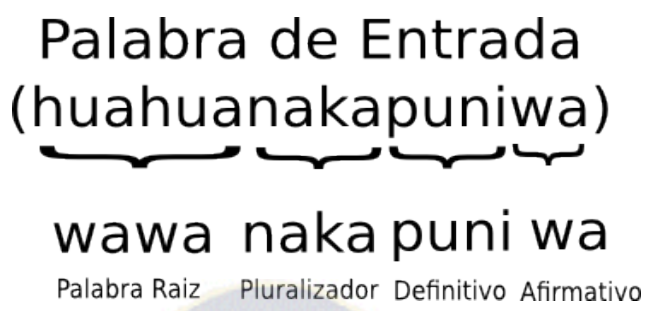

Figura 3.3: Composición de un Sustantivo. [Elaboración Propia]

Reglas de Corrección y de teclado: En este módulo se dará las reglas de convertir las vocales fuertes del español a las vocales del aymara, tambien se convertira consonantes del Espaõl al aymara, eliminación de marcas para los distintos módulos, que los sufijos no se repitan.

Reglas elisión vocalica: En este módulo se dará las reglas de elisión vocálica.

Reglas alargamiento vocalica: En este módulo se dará las reglas de alargamiento vocálica.

Búsqueda de palabra: Se realizará la búsqueda de palabra con instrucciones foma y mostrara las posibles correcciones de la palabra introducida en foma.

Palabra	Tipo	Posibles Correcciones
Achacu	Sustantivo	Achaku
Fusaña	Verbo	phusaña
Auquili	Adjetivo	Awkili
Yaquimaya	Adverbio	llakimaya

Tabla 3.1: Reglas de la palabra. [Elaboración Propia]

3.3. Variables del modelo

Las variables de entrada son los siguientes:

Palabra de entrada: palabra que se va a corregir.

Composicion de la palabra y reglas: sufijo o silaba para conformar diferentes combinaciones de palabras o con las que se descompone la palabra.

Las variables de salidas son:

Palabras de salida: palabra o similitud de palabras con las cuales se va a corregir.

3.4. Descripción formal del Autómata

El diseño del proceso de corrección de una palabra, se lleva acabo tomando en cuenta las características fundamentales de cada palabra.

Las palabras son comparadas y corregidas con autómatas en la verificación de cada palabra necesaria para poder elegir la regla de corrección adecuada de la palabra.

3.4.1. Función de la división de la palabra

Sea la función que se define:

$$X : T \rightarrow BL$$

donde:

T : es la palabra que se va corregir, entonces se divide la palabra en una palabra raíz y sus sufijos.

BL : es el proceso de búsqueda de la palabra.

X : Es la función de separación de la palabra.

3.4.2. Función de análisis de comparación

Sea la función de comparación definida como:

$$Y : BL \rightarrow SS$$

donde:

SS : es la proceso que realiza una comparación de las palabras.

Y : Es la función que usa para la comparación para poder procesar la palabra.

3.4.3. Corrección

Sea la función W definida como:

$$W : SS \rightarrow CC$$

donde:

CC : son las posibles palabras para corregida en el idioma aymara.

W : Es la función que obtiene la corrección de la palabra.

Gráfico de armado de la palabra que se detalla en la Figura **??**.

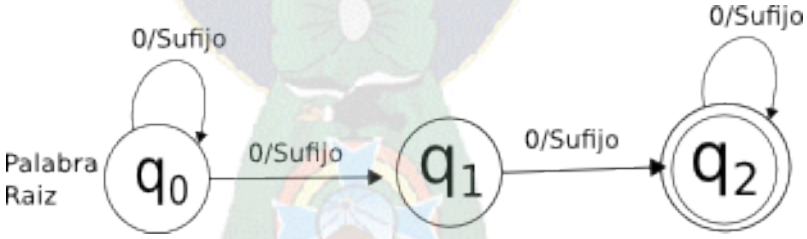

Figura 3.4: Armado de la palabra. [Elaboración Propia]

donde:

$Q = \{q_0, q_1, q_2\}$: Conjunto de estados.

$\Sigma = \{a, ch, chh, ch', j, k, l, ll, m, n, \tilde{n}, i, p, ph, ph', q, qh, qh', r, s, t, th, t', u, w, x, y\}$: Alfabeto de entrada.

$\delta : Q \times \Sigma \rightarrow Q$: Función de transición.

$Q_0 = \{q_0\}$: Estado inicial.
$F = \{q_2\}$: Conjunto de estados finales.

Este autómata recibe como entrada la palabra dividida en una palabra raíz y sufijos en un estado q_0, cuando encuentra el símbolo +/@ pasa a un estado q_1, si el sufijo pasa a armar la palabra en q_2

Autómata de un Verbo:

Se parte de una palabra raíz que es un Verbo el cual sera cargado en el archivo lex y tendra un orden con sufijos debidamente adicionados a la palabra raíz verbo el cual formara la nueva palabra verbo que se detalla en la Figura 3.5.

donde:
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17}, q_{18}, q_{19}, q_{20}, q_{21}, q_{22}, q_{23}, q_{26}\}$: Conjunto de estados.
$\Sigma = \{wisa, qata, li, lla, chu, ku, ita, itu, naka, ni, ja, ma, pa, sa, layku, pura, mpi, jama, ...\}$: Alfabeto de entrada.
$\delta : Q \times \Sigma \to Q$: Función de transición.
$Q_0 = \{q_0\}$: Estado inicial.
$F = \{q_{26}\}$: Conjunto de estados finales.

Autómata de un Sustantivo o nombre: Se parte de una palabra raíz que es Sustantivo o nombre el cual sera cargado en el archivo lex que se detalla en la Figura 3.6.

donde:
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17}, q_{18}, q_{19}, q_{20}, q_{21}, q_{22}, q_{23}, q_{24}\}$: Conjunto de estados.
$\Sigma = \{wisa, qata, li, lla, chu, ku, ita, itu, naka, ni, ja, ma, pa, sa, layku, pura, mpi, jama, ...\}$: Alfabeto de entrada.
$\delta : Q \times \Sigma \to Q$: Función de transición.
$Q_0 = \{q_0\}$: Estado inicial.
$F = \{q_{24}\}$: Conjunto de estados finales.

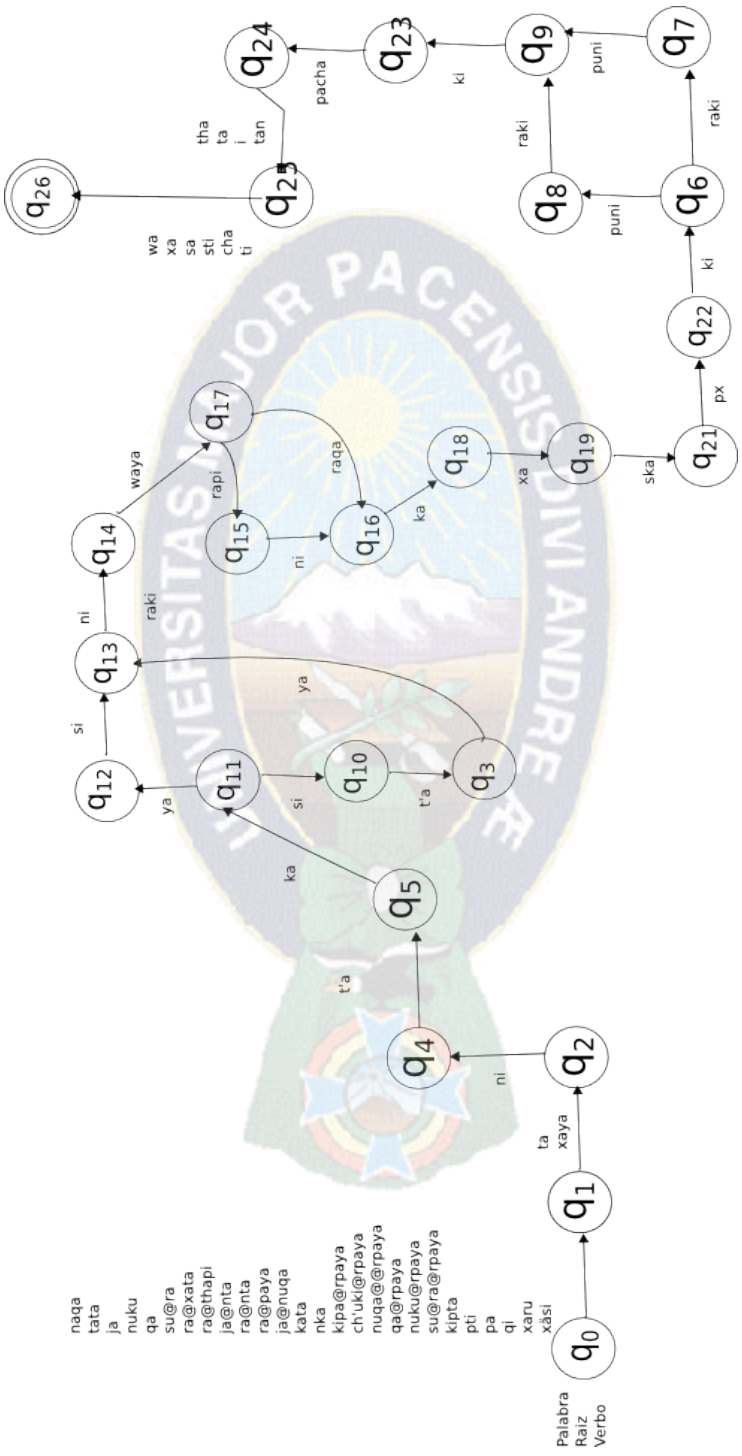

Figura 3.5: Autómata de Verbo. [Elaboración Propia]

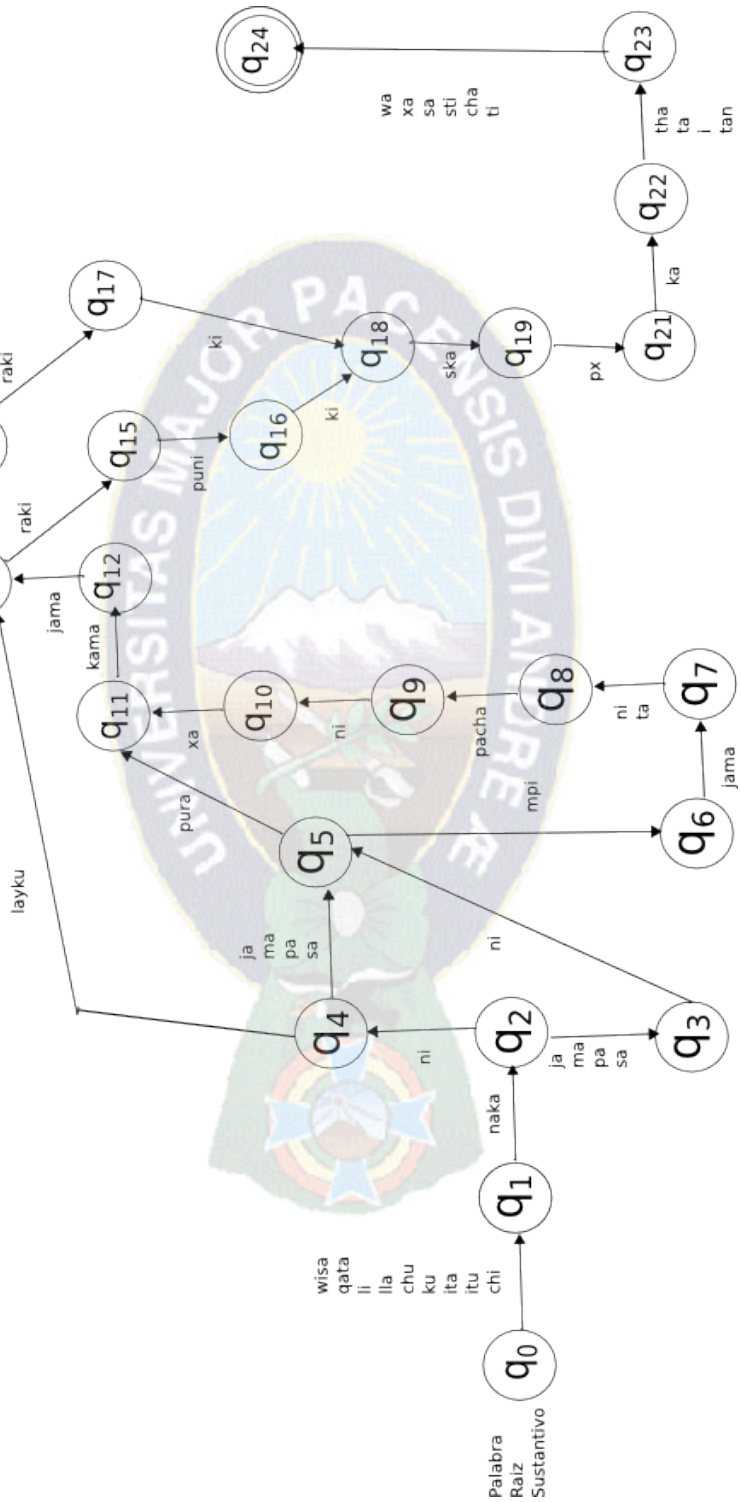

Figura 3.6: Autómata de Sustantivo. [Elaboración Propia]

3.5. Infraestructura

3.5.1. Software

Foma es compatible con los sistemas operativo Windows Xp, Vista, Siete y Ocho, y también es compatible con el sistema operativo Gnu-Linux cualquiera sea la distribución existente.

3.5.2. Hardware

El sistema requiere una computadora personal, donde el usuario pueda realizar las correcciones, Para el estudio se eligio el sistema operativo Ubuntu 12.10, con un procesador i5, con memoria RAM de 4 Gigas.

3.5.3. Funciones del software

FOMA se ejecutara en el Sistema Operativo Ubuntu 12.10 el cual usara librerias del lenguaje C para ejecutar los lex para corregir los errores en el Idioma Aymara, el cual estara compuesto por tres archivos que son:

- arsunakawa02.lex
- varianteteclado02.foma
- ayamara_corrector.foma

arsunakawa02.lex un archivo lex donde esta el diccionario de datos el cual esta dividido en Sustantivos, Verbos, Adjetivos, Adverbios, Pronombres y pronombres Interrogativos y la composición de la palabra raiz con los sufijos para cada uno de los casos mencionados en la Figura 3.7.

```
!!!##########################################################################
!!!Program: !!!arsunakawa02.lex!!!
!!!Author:  Edwin N. Saca Apaza, email: saca.nestor@gmail.com, web: www.illa-a.orgajra
!!!Created: 3 Jun 2014, (La Paz, Bolivia)
!!!###########################################################################
!!!Script de Sufijos para Sustantivos, Verbos, Adverbios, Adjetivos, Pronombres, Pronombres interrogativos en aymara
!!!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
!!!@@@@ Root                                                      @@@@
!!!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

LEXICON Root

Verbos;
Sustantivos;
Adjetivos;
Adverbios;
Pronombres;
PronombresInterrogativos;
!Pala;

!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
!@@@ Sustantivos                                                @@@@
!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
LEXICON Sustantivos

achachi[Sustantivo]:achachi CC_sustantivos;
achachila[Sustantivo]:achachila CC_sustantivos;
achakana[Sustantivo]:achakana CC_sustantivos;
achaku[Sustantivo]:achaku CC_sustantivos;
achalari[Sustantivo]:achalari CC_sustantivos;
achanqara[Sustantivo]:achanqara CC_sustantivos;
achika[Sustantivo]:achika CC_sustantivos;
achiku[Sustantivo]:achiku CC_sustantivos;
achila[Sustantivo]:achila CC_sustantivos;
achinu[Sustantivo]:achinu CC_sustantivos;
achira[Sustantivo]:achira CC_sustantivos;
achiwa[Sustantivo]:achiwa CC_sustantivos;
achu[Sustantivo]:achu CC_sustantivos;
achulla[Sustantivo]:achulla CC_sustantivos;
achuma[Sustantivo]:achuma CC_sustantivos;
achupalla[Sustantivo]:achupalla CC_sustantivos;
achupaya[Sustantivo]:achupaya CC_sustantivos;
```

Figura 3.7: Archivo arsunakawa02.lex [Elaboración Propia]

varianteteclado02.foma es un archivo que contiene las reglas de vocales y consonantes para refonetizar del idioma español al idioma aymara en la Figura 3.8.

```
define TecladoQU {qu} (<-) k | {k'} | {kh} | q | {q'} | {qh} || _ e | i ;

!define TecladoC c | C (<-) c | C | k | {k'} | {kh} | q | {q'} | {qh} ;

define TecladoK k | {k'} | {kh} (<-) k | {k'} | {kh} ;

define TecladoQ q | {q'} | {qh} (<-) q | {q'} | {qh} ;

define TecladoKQ {kh} (<-) {qh} ;

define TecladoCH {ch'} | {chh} | {ch} (<-) {ch'} | {chh} | {ch} ;

define TecladoP {p'} | {ph} | p  (<-) {p'} | {ph} | p ;

define TecladoF f (<-) {ph} ;

define TecladoT {t'} | {th} | t  (<-) {t'} | {th} | t ;

define TecladoL l | y (<-) {ll} ;

define TecladoW {hu} | {HU} | {Hu} | {gu} | {GU} | {Gu} (<-) w ;

define TecladoN n | ñ (<-) n ;

define TecladoX {jj} (<-) x ;

define TecladoAW a u (<-) a w,,
                 u (<-) ü ;
```

Figura 3.8: Archivo varianteteclado.foma [Elaboración Propia]

ayamara_corrector.foma es el que contiene las reglas las composiciones e elisiones y alargamientos vocalicos que se detalla en la Figura 3.9.

```
#Posesivos
define Consonantes [ c h | c h h | c h ' | j | k | k h | k ' | l | l l | m | n | ñ | p | p h | p ' | q | q h | q ' | r | s | t | t h |
define Vocales [ a | u | i | ä | ü | ï | á | ú | í | A | U | I | Ä | Ü | Ï | Á | Ú | Í ];
define Posesivos [ m | p | s | r ] ;
define Mayusculas [ c h (->) C H, c h ' (->) C H ', c h h (->) C H H, j (->) J, k (->) K, k h (->) K H, k ' (->) K ', l (->) L, l l (->
define Minusculas [ C H (->) c h, C H ' (->) c h ', C H H (->) c h h, J (->) j, K (->) k, K H (->) k h, K ' (->) k ', L (->) l, L L (->

#define EliVocal [ o (->) u, e (->) i, O (->) U, E (->) I ];
read lexc arsunakawa02.lex
!read lexc aru.lex
define LEX;

# Reglas

#Uso de sufijos problematicos
#####################################################################
#  Elisión morfémica                                                #
#####################################################################
#Sufijos extrafuertes (representado por #1)                         #
#Sufijos profuertes (representado por #2)                           #
#Sufijos semifuertes (representado por #3)                          #
#Sufijos contrafuertes (representado por #4)                        #
#Sufijos fuertes (representado por #5)                              #
#SUfijos Normales (representado por nada)                           #
#####################################################################

# Variantes
# Variante (challwa <-> chawlla), (qarwa <-> qawra), (wayra <-> warya)
#define Variante01 l l w (->) w l l,,
#                   r w (->) w r ,,
#                   y r (->) r y ;
# Variante (nayra <-> layra)
#define Variante02 n a y (->) l a y,,
#                   x (->) j,,
#                   c h (->) k,,
#                   m (->) n,,
#                   x (->) k;
# Variante (añathuya <-> añuthaya)
#define Variante03 {ñathu} (->) {ñutha},,
#                   {mp'} (->) {mp}
```

Figura 3.9: Archivo ayamara_corrector.foma [Elaboración Propia]

3.5.4. Foma ejecutando el corrector Ortográfico

Para ejecutar foma se necesita instalar el foma en el sistema operativo y escribir los siguientes comandos:

```
foma -l aymaracorrector.foma
```

Se ejecuta el archivo ayamaracorrector.foma el cual invoca a los otros archivos foma y lex, dentro de foma se genera un archivo de 6.8 MB, 9 trillones de palabras que se detalla en la Figura 3.10.

```
defined Rclean09: 268 bytes. 1 state, 2 arcs, Cyclic.
defined Rcleanmas: 268 bytes. 1 state, 2 arcs, Cyclic.
defined Rcleanarroba: 268 bytes. 1 state, 2 arcs, Cyclic.
defined Rclean: 568 bytes. 1 state, 12 arcs, Cyclic.
defined ReglasSinELision: 62.9 kB. 250 states, 3987 arcs, Cyclic.
defined ReglasConElision: 63.7 kB. 263 states, 4037 arcs, Cyclic.
defined AllRules: 189.3 kB. 772 states, 12077 arcs, Cyclic.
defined MorfoEstandar: 793.2 kB. 16593 states, 50687 arcs, more than 92233720368
54775807 paths.
defined MorfoVariante: 1.8 MB. 27821 states, 116006 arcs, more than 922337203685
4775807 paths.
defined Morfo: 6.8 MB. 23747 states, 443211 arcs, more than 9223372036854775807
paths.
6.8 MB. 23747 states, 443211 arcs, more than 9223372036854775807 paths.
Writing to file correctoraymara.fst.
Foma, version 0.9.16alpha
Copyright © 2008-2011 Mans Hulden
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; for details, type "help license"

Type "help" to list all commands available.
Type "help <topic>" or help "<operator>" for further help.

foma[1]:
```

Figura 3.10: Ejecución del Corrector. [Elaboración Propia]

Probando el corrector con la palabra auquili y nos muestra las distintas opciones de correción en la Figura 3.11.

```
foma[1]: up
apply up> auquili
awkili
awkilit
awkilita
awkilitayn
awkilitayna
awkilist
awkilista
awkill
awkilli
awkilltayn
awkilltayna
awkil
awkiltayn
awkiltayna
```

Figura 3.11: Corrección de la palabra auquili. [Elaboración Propia]

Capítulo 4

EVALUACIÓN DE RESULTADOS

4.1. Análisis de datos y resultados

La corrección es difícil, se requiere de un profundo conocimiento de la palabra y para ello es necesario tener un conocimiento más amplio del idioma Aymara. Es válido para palabras simples o como para palabras compuestas por varios sufijos.

Este trabajo fue desarrollado en base a los principales elementos de la gramática Aymara, en las que se basa la corrección de las palabras.

La verificación de las estructuras de las palabras fue desarrollado en base a los autómatas finitos.

La operacionalización de variables de una herramienta es muy importante, con el cual verificamos la veracidad de nuestra hipótesis.

Para la verificación de la corrección correcta, se hizo pruebas del prototipo en la corrección de las palabras, los datos resultantes están agrupados en la siguiente tabla.

Donde:

- Nro. de palabras correctas reconocidas.
- Nro. de palabras que se pueden corregir.
- Nro. de palabras que no se pueden corregir.
- Nro. de palabras corregidas.
- Valor: de la corrección con fórmula.
- Complemento del valor: es el resultado del complemento del valor.

$$Corrección = \frac{Número\ de\ palabras\ corregidas}{Número\ de\ palabras\ que\ se\ pueden\ corregir}$$

$$PROMEDIO_{Corrección} = \frac{\sum_{i=1}^{n} VALOR_i}{n}$$

Para la prueba se escaneo textos antiguos del idioma Aymara con varios errores ortograficos, y se conto con especialista Jose Laura Yapita que es egresado de la carrera de Lingüistica del Idioma Aymara de la Universidad Mayor de San Andres y sobrino de Juan de Dios Yapita un reconocido Lingüistica Aymara, para la verificación de la corrección de los errores ortográficos.

Los textos que se usaron los siguientes libros para comprobar los errores, son los siguientes

- Aaymara Lupiwisa Arusa Amuyasa de Zacarias Alavi.
- Canticos aymaras.
- Layra Parla de Braulio Choque Mamani.
- Literatura Aymara
- Literatura Aymara y Cuentos Andino

4.2. Datos obtenidos del corrector ortográfico

Se obtuvo los siguientes datos con el corrector ortográfico

Pruebas	Palabras de entrada	Palabras reconocidas	Palabras que se pueden corregir	Palabras corregidas	Valor
1	102	86	15	11	0.73
2	57	49	8	6	0.75
3	175	140	20	11	0.55
4	58	40	10	8	0.8
5	89	69	21	12	0.57

Tabla 4.1: Diccionario de datos del corrector [Elaboración propia]

4.3. Datos obtenidos del lingüista

Se obtuvo los siguientes datos del lingüista Aymara.

Nro.	Nro. de palabras de entrada	Nro. de palabras reconocidas	Nro. palabras que se pueden corregir	Nro. palabras corregidas	Valor
1	102	86	15	13	0.86
2	57	48	8	7	0.87
3	175	140	20	15	0.75
4	58	40	11	9	0.81
5	89	69	20	12	0.6

Tabla 4.2: Diccionario de datos del lingüista Aymara[Elaboración propia]

Con todos los datos anteriores hallamos otra tabla para 5 muestras del corrector ortografico.

Pruebas	Palabras corregidas por el lingüista	Palabras corregidas por el corrector
1	13	11
2	7	6
3	15	11
4	9	8
5	12	12

Tabla 4.3: Datos unidos del lingüista y el corrector [Elaboración propia]

$$n_1 = 5 \; y \; n_2 = 5$$

$$x_1 = \frac{13+7+15+9+12}{5} = 11,2$$

$$x_2 = \frac{11+6+11+8+12}{5} = 9,6$$

Hipótesis:

H_i: $u_0 = u_1$"El corrector ortográfico con autómatas finitos usando **FOMA** permite corregir las palabras con errores ortográficos, del idioma Aymara con un reconocimiento de palabras de hasta un 70 % ".

H_O: $u_0 \neq u_1$"El corrector ortográfico con autómatas finitos usando **FOMA** no permite corregir las palabras con errores ortográficos, del idioma Aymara con un reconocimiento de palabras de hasta un 70 % ".

Se ha realizado 5 experimetos para estudiar el comportamiento del corrector ortografico.

- Dando un α

$$\alpha = 0{,}05$$

- Region de rechazo $1-\alpha = 0,95 \rightarrow t_{(8,0,95)} = 1{,}8595$, o sea $R.C. =< 1{,}8595, \infty >$

- Puesto que $n = 5$ es pequeño usaremos la distribución **t**. $S^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$
- Cálculo de $\bar{x} = 13$, $n = 5$.

$$S_1^2 = \frac{\sum_{i=1}^{5}(x_i - \bar{x})^2}{n-1}$$

$$= \frac{(13 - 11,2)^2 + (7 - 11,2)^2 + (15 - 11,2)^2 + (9 - 11,2)^2 + (12 - 11,2)^2}{5-1}$$

$$= 10,2$$

$$S_2^2 = \frac{\sum_{i=1}^{5}(x_i - \bar{x})^2}{n-1}$$

$$= \frac{(11,2 - 9,6)^2 + (6 - 9,6)^2 + (11 - 9,6)^2 + (8 - 9,6)^2 + (12 - 9,6)^2}{5-1}$$

$$= 6,3$$

$$t_c = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$t_c = \frac{11,2 - 9,6}{\sqrt{\frac{(5-1)10,2+(5-1)6,3}{5+5-2}(\frac{1}{5} + \frac{1}{5})}}$$

$$= 0,88$$

- Conclusión ya que la $t = 0,88$ no esta en la región critica no se rechaza H_0, por tanto se acepta.

La hipotesis se acepta con $\alpha = 5\,\%$, es decir el procentaje es de $95\,\%$ para el corrector ortográfico.

Capítulo 5

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

El trabajo realizado ha sido planteado en un procedimiento de autómatas finitos para realizar el analisis morfológico de una palabra y posibles combincaciones de sufijos y conformara otras palabras.

Con el desarrollo del corrector se ha buscado que la corrección sea correcta, es decir las palabras o opciones de corrección debera ser inteligible y representa la palabra correcta.

Se implemento el diccionario de palabras, laconformación de otras palabras en base al diccionario con los sufijos para cada caso.

Las reglas de corrección de teclado, las elisiones y alargamientos vocalicos.

El análisis morfologico nos permite categorizar las palabras en sub grupos para su correspondiente estudio. se desarrollo un corrector para la corrección del idioma aymara.

5.2. Recomendaciones

Hoy en dia es muy necesario aprender este lenguaje, para poder escribir y comunicarse con los hablantes aymaras, ya que el Estado Plurinacional de Bolivia esta priorizando las lenguas nativas del Pais.

Las recomendaciones son las siguientes:

Buscar otros diccionarios a parte del que se uso en el trabajo, para realizar una conjunción de estos y así de esta manera enriquecer la lengua ayamra.

Llevar a un procesador de textos y otras tecnologias.

Bibliografía

[1] Teofilo Layme Ajacopa. *Como escribir en Aymara*. Vision, 2012.

[2] Google Foma. Getting started foma. http://code.google.com/p/foma/wiki/GettingStarted, 2014. [Accesado Marzo-2014].

[3] Ullman Hopcroft. *Introducción a la teoria de Autómatas lenguajes y computación*. Compañia Editorial Continental, 1993.

[4] Félix Layme Pairumani. *Diccionario Bilingüe Aymara castellano - Castellano aymara*. CEA, 2009.

[5] Correctores Ortgráficos para multiples idiomas. Ortográfias endometricas. http://eventos.spc.org.pe/cicp2005/papers/0054/ortografico_edmetricas.pdf, 2014. [Accesado Marzo-2014].

[6] Flores Ramiro. *Apuntes de teoría de automatas*, 2010.

[7] Marline I. Vedia Rodriguez. Inteprete español - quechua. 2002.

[8] Jesus Carretero Santiago Rodriguez. Corrector ortografico de libre distribucion basados en reglas de derivacion. http://www.datsi.fi.upm.es/coes/publications/Eguth.pdf, 2014. [Accesado Marzo-2014].

[9] Teodosia Paye Ticona. Traducción semántica y sintáctica del idioma español al aymara utilizando autómatas. 2012.

Anexos A

Textos para la corrección

A.1. TIWULANPI HUALLATAMPI

Nayra pachax ma huallataw wawanakapampi kuta thiyan lap'arasisipkän siwa, Uka chhiwchhinakapax, k'illu sank'ayjamanwa siwa.

Yakhachiqansti, mama tiwulax wali willjtataraki utjawipat wawanakapatak mank'a thaqir Jalsuniwayatayna wich'inkhas tantat jalaskan siwa.

Ukham jalkas jalkas tiwulax mama wallatampiw thakhin jikthaptatayna, Ukat tiwulax akham sataynaw. !Ay! mama wallata, kunatrak wawanakamax akha suma paruk paruki, wali jayarpachaw qurjam k'ajtanitu sasa.

Ukham siri istasax mama wallatax wawapanakampix walrak mulljasipxatayna. Ukat wallatax jiskt'atayna, wawanakamasti janit ukhamanakakisti, mama tiwula sasaw satayna Mama tiwulax wali llakt'asit chuymapampiw jaysatana, janiw wawanakajax ukham jiwanakakïti mama wallata sarakitaynawa Mama wallatax ukam ist'asax ist'asax jaypachaw musparpayarakitayna, chiwchinakapasti jahuir thiyanraki sillq'unak pallasisipkan siwa.

Ukat mama tiwulax satayna !Ay! mama wallta kunjamatpunrak wawanakamasti ukham paruk parukisti, janit yatichitasma sasaw satayna.

Ukat mama wallatax sarakitayna, Nayraqatxa kulanak pirksuta, ma wajañ utjam, ukat ukaruw wali parintañapkam tulanakampi phichata, Ukat ukaruw wawanakam allintäta ukat phunk! Sanixa ma qillqa sasaw muytata , mayampi phunk sisxarakikinixa pa qillqa satawa ukxarux mayamp !phunk! sisxarakikinixa kimsa qillqa satawa

ukat mama tiwula wawanakamx paruk paruki apsxata sasaw yatichatayna.

Ukat mama tiwulax wawanakap p'urkharañatakix wali kusisitaw mä urux khulanak pirkasitayna asta puniw phichhatayna, wallatan yatichawiparjam. Niya nina jiwaraskirirjamarux wawanakap antanitayna uka waja utar p'urkharañataki ukat mama tiwulax akham sataynawa, jichhax wawanakajax wali jiwitanakakïxaniwa sasa.

A.2. TIWULAMPIT WAYRAMPITA

Tiwulampit wayrampita wayra anturpayxita ayapachanxa tiwulaw wayrar ikkirir jikxattatayna ukat akham satayna kunats jumax wäli qhusquk ikisjta wayrax janiw kuns ist'kataynati ukhamaw tiwulax t'awra tukusitayna mayax manqha luma tanisanw qhusqhutatha, sasaw wayrax tiwular itayna ukaruw aka siw sawix tukuyi katx tiwulax sarakitaynawa mayar irpita sasa mallkunakamp lulinakamp nuwasitapxata mä uruxa mallkuw qawr manq'antatayna jupax sintipuniw sisantxatayna inti sisantxasinsti janiw jaltañ atxataynati ukat taqi laq'u masinakaparuw iwsasitayna inaw qásaskatayna siwa khamä kipanx mä jan yäqat lulikiw purinitayna tata mallkux akham rxayatayna qun wawa purakaw wälch'itintxitu ikanx lulix inamayakiw misk'i uma mallkur umayatayna, siwa ukat mallkux lulir akham satayna jila masinakañ arjawstánirapita mpi.

A.3. KUNA ASKIRAK

kuna askirak jaqisti amtasiñamatak munat wawam katuyasta naya juchan layku alaxpach parki uñtasax phaxsimax musphayituwa suma dios awkixa intis wara waranakas luratam yatiyasipki qullan dios awkixa kunja jachás sutimaxa taqi akapachan kunja sumas luratamax taqi jaqitaki nayasti akapachana luratanakam uñjasax jumar arkañ munta markamar puriñaxkama yanaptítay dios awkixa juman wawamätwa amtasiyitay awkixa qullan sutimata yatichasitay awkixa arunakamata.

A.4. PÁXPÁKUBT KÚSILUJNTI

Páxpákunt kúsilujnti má jaqint má kúsilluntix qhathut qhath markat mark sarapxiritayna uka kúsillurux jaqinakatakiw surtill apsurapiritayna pampachaniw suma, jan chúxñasis qamasipxatayna ukatx jupanakatakix pistáñ urunakaw purinitayna janiw lurañas ni qullqis walj utjxataynati ukatx jachá markar sarxañán sasaw sarxapxatayna taqikunas kháns ukans kipkakirajitaynawa ukatx jiská marka qhathunakaruw kuttápxatayna wari wali jachá qhathuruw puripxatayna ukanx má urux ñakakiw manqáñatak qullqix utjatayna uka tatant kúsilluntix chúxña juwas manqásipxatayna uka kúsillux n yat niyat tatar amparap juwas churañpatak aykatiritayna walja juwas manqántapxatayna mawkántjitarux juwasax tukusxataynawa kusillux amparap uka tatar luqkataskakitaynawa janiw chu'xña juwasax utjxituti, qálaxiwa sasaw satayna kúsillux janiw tatar ist'kataynati ukatx uka tatax kamacht'ás sataynawa janiw kuna qullqis chúxña juwas alañatak utjxataynati sismawá, janiw chúxña juwasax utjxituti qálaxiwa janiw ni qullqis alañatak utjxituti mayant satayna kúsillux janiw ist'kataynati, amparap luqkataskakitaynawa uka jaqix jan kamachtáñ atxasax wulsikupan juwas thaqhatayna chúspapanx chúxña luqutuk katutayna, uk churatayna kúsillux sum manqánttásitayna, chúxña juwasákaspas ukhama mawkántjitarux luqutux wál jaruntxatayna ukatx kúsillux nayrap jachá jistárasitayna nayrapax thuqsuñ munatayna ukham uksar aksar chírminaqatayna lakaps ansaqiyarajitaynawa luqutuw ukjam thuqnaqayatayna uka tatax janiw kunjams kúsillur qulltáñ atkataynati.

Anexos B

Tabla t-student

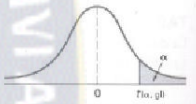

TABLA DE LA DISTRIBUCIÓN t de Student.

Para un número particular de grados de libertad, las entradas representan el valor crítico de t correspondiente a un área de extremo superior especificada (α).-

Grados de Libertad	AREAS DE EXTREMOS SUPERIOR (α)					
	0.25	0.10	0.05	0.025	0.01	0.005
1	1.0000	3.0777	6.3138	12.7062	31.8207	63.6574
2	0.8165	1.8856	2.9200	4.3027	6.9648	9.9248
3	0.7649	1.6377	2.3534	3.1824	4.5407	5.8409
4	0.7407	1.5332	2.1318	2.7764	3.7489	4.6041
5	0.7267	1.4759	2.0150	2.5706	3.3649	4.0322
6	0.7176	1.4398	1.9432	2.4469	3.1427	3.7074
7	0.7111	1.4149	1.8946	2.3846	2.9980	3.4995
8	0.7064	1.3968	1.8595	2.3060	2.8965	3.3554
9	0.7027	1.3830	1.8331	2.2622	2.8214	3.2498
10	0.6998	1.3722	1.8125	2.2281	2.7638	3.1693
11	0.6974	1.3634	1.7959	2.2010	2.7181	3.1058
12	0.6955	1.3562	1.7823	2.1788	2.6810	3.0545
13	0.6938	1.3502	1.7709	2.1604	2.6503	3.0123
14	0.6924	1.3450	1.7613	2.2448	2.6245	3.9768
15	0.6912	1.3406	1.7531	2.1315	2.6025	2.9467
16	0.6901	1.3368	1.7459	2.1199	2.5835	2.9208
17	0.6892	1.3334	1.7396	2.1098	2.5669	2.8982
18	0.6884	1.3304	1.7341	2.1009	2.5524	2.8784
19	0.6876	1.3277	1.7291	2.0930	2.5395	2.8609
20	0.6870	1.3253	1.7247	2.0860	2.5280	2.8453
21	0.6864	1.3232	1.7207	2.0796	2.5177	2.8314
22	0.6858	1.3212	1.7171	2.0739	2.5083	2.8188
23	0.6853	1.3196	1.7139	2.0687	2.4999	2.8073
24	0.6848	1.3178	1.7109	2.0639	2.4922	2.7969
25	0.6844	1.3163	1.7081	2.0595	2.4851	2.7874
26	0.6840	1.3150	1.7056	2.0555	2.4786	2.7787
27	0.6837	1.3137	1.7033	2.0518	2.4727	2.7707
28	0.6834	1.3125	1.7011	2.0484	2.4671	2.7633
29	0.6830	1.3114	1.6991	2.0452	2.4620	2.7564
30	0.6828	1.3104	1.6973	2.0423	2.4573	2.7500
31	0.6825	1.3095	1.6955	2.0395	2.4528	2.7440
32	0.6822	1.3086	1.6939	2.0369	2.4487	2.7385
33	0.6820	1.3077	1.6924	2.0345	2.4448	2.7333
34	0.6818	1.3070	1.6909	2.0322	2.4411	2.7284
35	0.6816	1.3062	1.6896	2.0301	2.4377	2.7238
36	0.6814	1.3055	1.6883	2.0281	2.4345	2.7196
37	0.6812	1.3049	1.6871	2.0282	2.4314	2.7154
38	0.6810	1.3042	1.6860	2.0244	2.4286	2.7116
39	0.6808	1.3036	1.6849	2.0227	2.4258	2.7079

Anexos C

Documentación