UNIVERSIDAD DE BUENOS AIRES

Reconocimiento y Corrección de Errores por Ingreso de Lenguaje Natural en Dispositivos Electrónicos

Tesis de grado

INGENIERIA ELECTRONICA

Autor

Andrés Tomás Hohendahl, andres.hohendahl@fi.uba.ar

Director

Dr. José. F. Zelasco, jfzelasco@fi.uba.ar

Co-Director

Dr. Ing. Silvano B. Zanutto, silvano@fi.uba.ar

Año 2014

Agradecimientos

Al director de esta tesis, José Zelasco y al codirector Silvano Zanutto, por saber como guiarme en un trabajo complejo, con sus sabios y alentadores consejos. A José, en especial por su infinita paciencia, acogiéndome en su casa innumerables veces para revisar la redacción palmo a palmo, enseñándome no solo a cómo pensar científicamente, sino también el cómo comunicarlo en forma clara, simple y amena.

También le agradezco a esta facultad de ingeniería y su abnegado conjunto de docentes; a los buenos por regalarme su visión, tiempo y paciencia, contagiándome la pasión por las ciencias que enseñaban; aunque también les agradezco a los malos, pues ellos me enseñaron como solucionar y saltear contingencias, incluyéndolos. Sin ambas cosas jamás me sentiría completo en mi formación.

Al Ing. Luis Rocha que no está más entre nosotros, pero fue un faro en mi vida y todo un ejemplo de ingeniero y por sobre todo una excelente persona, habiendo sido mi jefe por casi 10 años en el Instituto de Ingeniería Biomédica. Tengo que reconocer también a un grupo que considero un ejemplo a seguir como ingenieros como Jorge Alberto, Alberto Dams, Eduardo Mariani, del Giorgio, Álvaro López, Daniel Sinnewald, Bruno Chernuchi, Luis Marrone y otros muchos, que seguro me estoy olvidando en esta lista.

A mis padres el darme la chispa de la vida y legarme con sus genes lo que soy hoy: un emprendedor apasionado e incansable. En especial a mi padre quien ya no está, y me legó un ejemplo de sabiduría y sus fuertes ganas de ser ingeniero; también a mi madre de quien heredé su talento y pasión por la matemática, quien siempre me alentó a perseguir mis sueños, no importando el esfuerzo.

A mi adorada esposa Alejandra por tenerme la infinita paciencia que se debe tener con los apasionados y entusiastas, adictos al trabajo y la investigación en ciencias duras; y en especial, por no haberme echado aún de casa, por todo esto.

---

1 Introducción

En este capítulo, se presentan primeramente los objetivos de esta tesis, con una visión resumida de lo que se desea obtener, a continuación se presentan las especificaciones propuestas y la aplicación práctica estimada; completándose finalmente con una descripción de la organización del presente documento, para obrar de guía de lectura.

1.1 Objetivos

Desarrollar un modelo teórico-práctico para corregir errores y robustecer el ingreso de texto en lenguaje natural en dispositivos electrónicos con una apropiada identificación, detección y restauración de los errores, tanto ortográficos como tipográficos.

Se propone realizar el trabajo para el idioma español, por ser especialmente complejo, debido a su amplia difusión (~350.10^6 hispano-parlantes/2010) y por haber hallado escasos desarrollos en el estado del arte a la fecha de comienzo de este trabajo, que solucionen o se aproximen a una solución razonable del problema.

Una vez lograda la implementación, se aplicaron diversas métricas para evaluar si se cumplía los objetivos propuestos y hacer un comparativo con el estado del arte.

1.2 Especificaciones

Se propone implementar, probar y medir un conjunto de métodos que permitan analizar un texto, desmenuzarlo en constituyentes útiles y *detectar+corregir* el error tanto ortográfico como de tipeo, dicción u otro desconocido.

Deseamos que la reparación sea de una forma lo más similar posible a la que haría un ser humano promedio no experto en el tema.

La tarea de restauración léxica, deberá ser realizada en forma no-supervisada (*desatendida*), entregando suficiente información a una etapa posterior de procesamiento, que reciba datos útiles para tomar las decisiones correctas con un mayor grado de inteligencia y resolver posibles ambigüedades que aparezcan.

Se busca proveer información completa y normalizada, a la salida, informando los problemas hallados y el tenor de las decisiones tomadas, a fin de facilitar la toma de decisiones en posteriores etapas.

Metas del Diseño

a) Corrección de errores robusta y de calidad similar a la de un humano, tratando de mejorar el estado del arte, con una métrica simple y clara.

b) Implementación en un lenguaje moderno y portátil, que corra en plataformas tanto móviles como, de escritorio y de servicios; con el uso mínimo de recursos de almacenamiento, tanto estático como dinámico.

c) Velocidad de respuesta superior a la de ingreso de datos de un humano en un teclado (>*200 palabras por minuto*), para poder asistir en tiempo real tanto a la escritura como a una comunicación en vivo.

1.3 Aplicación Práctica

La idea es desarrollar una alternativa de tecnología especializada, para dotar a los nuevos equipos electrónicos inteligentes, inclusive aquellos con hardware embebido, de un ingreso de texto más robusto, mejorando la calidad en la comunicación y diálogo hombre-máquina en lenguaje natural, cualquiera sea el método de ingreso: teclado numérico, alfanumérico, táctil, óptico, etc.

Se estima que este desarrollo se aplicará principalmente en las áreas de la electrónica de consumo masivo como los nuevos electrodomésticos inteligentes en domótica, robótica, productos portátiles multifuncionales tipo celulares, tabletas gráficas, cámaras, además de toda clase de instrumentación y equipos por venir.

Otro lugar de aplicación interesante del presente trabajo en motores de búsqueda como Google, Yahoo y en especial en buscadores más específicos de Bibliotecas, Centros de Información u otros. Allí el ingreso de texto lo realiza un humano y por lo general las personas no siempre saben a ciencia cierta como se escribe o deletrea lo que están buscando, en especial si es un apellido o una palabra o sigla poco conocida o frecuente.

Además las bases de datos actuales no pueden buscar bien por índices ‘aproximados’, en consecuencia es deseable un mecanismo de restauración léxica previo, por medio del cual un algoritmo inteligente ‘trate’ de detectar si hay errores de algún tipo y en ese caso transformar lo *mal* escrito en *la alternativa más viable*, pero existente en las bases.

1.4 Organización del Documento

Se analizan y enuncian primeramente las disciplinas necesariamente involucradas haciendo un análisis pragmático del estado del arte a la luz las necesidades observadas.

Se describen luego en detalle el problema de los errores de texto y sus antecedentes y fundamentos.

En los capítulos subsiguientes se presentan paulatinamente los pasos realizados, con un análisis pormenorizado del problema y su posible solución. En cada paso se realiza una identificación del subproblema a solucionar, haciendo un análisis comparativo del estado del arte y finalmente se explica la implementación decidida, justificando cada decisión y evaluación realizadas en cada paso.

El resto del documento describe, en etapas, el desarrollo de la investigación junto con la implementación del presente trabajo, presentando luego una batería de mediciones y resultados que permitirán finalmente discutir el aporte realizado y sacar las conclusiones pertinentes.

Finalmente se presentan las perspectivas de desarrollo a futuro para la posible continuación del presente trabajo.

En las Referencias se podrá hallar todo el material bibliográfico citado.

Se agregaron varios anexos para aclaraciones, no incluyendo este material no esencial para evitar alargar el desarrollo de la propia tesis y perder el foco.

Incluimos al final un glosario, para evitar dudas y facilitar la comprensión de algunos términos, abreviaturas o acrónimos no convencionales en ingeniería electrónica.

---

2 Desarrollo

En este trabajo, y a lo largo de las siguientes secciones, vamos a adentrarnos de a poco en la vasta temática del texto y el lenguaje natural, simplemente para explicar con que elementos, teorías y algoritmos se trabajó; a efectos de justificar la razón de su selección, mencionando sólo en los casos que se consideren necesarios, cuál fue el algoritmo o teoría seleccionado y/o involucrado o cuál es el estado del arte asociado.

Para no alargar innecesariamente este trabajo, se darán referencias bibliográficas solamente en las secciones y partes en donde se haya considerado que sustenten y/o fundamenten una toma de decisiones crucial del presente trabajo, siempre que no sean de la corte del público conocimiento en ingeniería electrónica.

La idea es que el presente trabajo sea conciso, ya que puede fácilmente parecer que se halla rozando la lingüística y de hecho se está convencido que el problema de los errores de texto son un problema duro de resolver, aún no resuelto en su totalidad; en el cual la ingeniería posee herramientas útiles para intentar arribar a una solución óptima.

Vamos a abordar el problema del texto con errores como un problema abstracto de comunicaciones y atacarlo con esas herramientas que hemos aprendido durante la carrera de *ingeniería electrónica*, incluyendo la inteligencia artificial y aprendizaje automático lado a lado con las teorías de información y comunicaciones; evaluando en cada caso la aplicabilidad y ventajas de cada método, en caso de poder aplicarlo.

2.1 Teorías Existentes

Las teorías que se ensayarán serán la de la teoría de la información pasando por la empírica-lingüística de *Zipf*[1], pasando por *Shannon* [31] y algunas posteriores.

Se usarán la teoría de errores en comunicaciones y su estadística, el reconocimiento de patrones de caracteres [11], aprendizaje automático de inteligencia artificial [9], empleo de estadísticas relacionadas con las frecuencias de aparición de segmentos de letras en corpus (*n-gramas*), aplicando estimación bayesiana, teoría de grafos y algoritmos para optimización en búsquedas [6] y estructuras óptimas [5] para reducir uso de recursos.

Finalmente, se escogerán los formatos lingüísticos de etiquetado, conjuntamente con la modelización del lenguaje como código de comunicación robusta, lematización y flexión [4], conjuntamente con su teoría de generación y se abordará la percepción humana del habla para presentar métricas capaces de estimar la similitud perceptiva de formas escritas [16].

Se presentará también un conjunto de mecanismos adicionales, necesarios en el campo y relacionados con elementos llamados entidades nombradas, de difícil reconocimiento.

[1] Ley empírica de Zipf, formulada en la década del 1940 por George Kngsley Zipf, lingüista de Harvard, según la cual en una lengua la frecuencia de aparición de cada palabra sigue una distribución aproximada a $1/n^a$ donde n es la posición de la palabra n-ésima y a es próximo a 1. (*de Wikipedia*)

2.3 Métricas

Sin duda las métricas son cruciales para toda evaluación, en especial cuando la misma no es estándar, en cuyo caso se deben seleccionar y tal vez hasta diseñar métricas apropiadas; dado que si esto no está bien hecho, todo el trabajo puede no ser evaluable.

Por eso se buscarán y escogerán métricas basadas en el problema a solucionar y aplicables a los modelos, junto con su estadística.

Se tratará de medir la pertenencia de un texto a un idioma; la dificultad con la que un texto o palabra sea pronunciable; se buscarán métricas que nos indiquen si una palabra es algo gramatical, un nombre o simplemente texto-basura.

Se desarrollarán métricas para poder medir y estimar el error de un texto, basado en parámetros como la fonética [1] y la estadística de su secuencia de letras.

Se emplearán modelos estadísticos con datos extractados de los diccionarios existentes junto con las reglas de ortografía y escritura [3], también se usará la formación de palabras conforme a sus teorías básicas de corte netamente lingüística [2].

Usaremos métricas y reglas aplicables al idioma español [2], el cual resulta altamente flexivo y derivativo, formando un sinnúmero importante de palabras, incluyendo un gran número de palabras 'inventadas' a diario, llamadas *parasintéticas*[2] [12], resultando en una enorme variabilidad y riqueza expresiva, por cierto muy compleja de procesar.

Se proponen estructuras con algoritmos eficientes para la búsqueda y almacenamiento del tipo descripto en [5] y [6], técnicas de inteligencia artificial [9], infraestructura para ingeniería de textos [10] y modelos de lenguaje [8] usados en procesamiento del habla.

Nota: Un tema que no se abordará explícitamente en esta tesis es la corrección basada en interdependencia semántica y contextual entre palabras; puesto que la complejidad y el espectro posible de teorías aplicables al respecto, exceden largamente los tiempos y recursos aplicables para una tesis de grado, además de pertenecer mayormente a un territorio propio de lo lingüístico, un nuevo terreno interdisciplinario llamado *Procesamiento de Lenguaje Natural* y con connotaciones orden psicológico, rozando lo cognitivo y perceptivo.

---

[2] Parasintética: se dice así a las palabras que se forman mediante cambios morfológicos de derivación o flexión y/o con agregado de prefijos y sufijos; estas palabras no pertenecen a diccionario alguno, pero son perfectamente comprensibles y correctas en un contexto dado. Las ciencias naturales, biológicas, médicas químicas y farmacéuticas son eximios acuñadores de ellas en gran cantidad, formando palabras kilométricas de difícil comprensión si no se es de la rama, su longitud llega hasta las 45 letras.

3 Antecedentes

En esta sección se hará un discurso pragmático de los antecedentes vinculados al desarrollo de la tecnología electrónica y de computación a lo largo de las últimas décadas para luego analizar el estado del arte involucrado y mostrar la problemática que se pretende solucionar en esta tesis. No nos adentraremos en temas que no resulten de aplicabilidad directa.

La Tecnología

La tecnología electrónica ha traído consigo un sinnúmero de cambios en los últimos tiempos. Hoy es casi impensable un dispositivo sin algún componente electrónico. Las comunicaciones, los medios masivos como la radio, la televisión, internet, satélites, microondas, láser, fibras ópticas, etc.; rigen, mueven y conectan hasta lo impensado del mundo de hoy.

El Avance

La penetración de pico, micro y mini computadoras en sus diversas formas en la vida cotidiana es cada día mayor, no es algo que se vaya a detener. Ya sea en forma de un reloj, un juguete (*gameboy, playstation*), un celular, tablet, note/net/ultra-book o una desktop PC, un servidor, el lavarropas o la heladera inteligente y tantos aparatos más.

Tarde o temprano habrá numerosas computadoras por persona, los dispositivos más impensados serán inteligentes y se comunicarán con nosotros o entre sí, conformando pico-redes para mejorar nuestro confort, eficiencia, seguridad y en definitiva contribuyendo con la calidad de vida, o no.

La Inteligencia Artificial

En las primeras películas futuristas de ciencia ficción, las computadoras imaginadas ya *"pensaban"* y tomaban decisiones, con caras de lata inexpresivas, un show de cintas de grabación girando con luces destellantes, a modo de ojos. No pocas veces en estas ficciones, se ha cuestionado éticamente el tema, planteándose el dilema casi bizantino de la *'inteligencia de las máquinas'* versus la del hombre.
Todavía continúa el dilema si algún día próximo o no, la inteligencia humana será superada por las máquinas, llamando a este evento: "la singularidad" por algunos futurólogos. Otros predicen que los humanos nos uniremos a la tecnología, solo el tiempo lo dirá.

Un simple Olvido

El crecimiento de los últimos 50 años fue realmente fabuloso, excediendo tal vez lo que podían haber soñado o predicho muchos visionarios, pero algo parece haber fallado en la interfase hombre-máquina, veamos a que nos estamos refiriendo.

En aquellas películas, se mostraban los operarios especializados presionando unos enromes botones con luces en las computadoras, para que hagan algo. Esto no cambió en 50 años, y hoy mismo, para sentarse a trabajar con una computadora, es impensable no tener que presionar botones en los teclados y/o superficies sensibles 'touch'. Lo

tristemente cierto, es que debemos ser mecanógrafos un tanto expertos, si necesitamos ingresar datos a una computadora sin ser un hazmerreír. El tema nada trivial que ponemos en foco ahora: es el mecanismo de ingreso y los errores que comentemos.

Los Teclados Actuales

Los teclados de hoy día, solo han evolucionado en sus materiales y estética, pero no distan mucho de los de las máquinas de escribir de antes *(cuyo tamaño, cada vez menor, responde tal vez a razones ergonómicas y/o de mercado).*

Usabilidad

Además no basta con mecanografiar una carta, hoy se debe conocer la existencia y funcionamiento unas cosas raras llamadas "programas", se debe entender sobre un tipo de ventanas que aparecen en una pantalla y no en la pared, se debe saber dar algunas órdenes extrañas e in-entendibles en el teclado, usando textos un tanto crípticos a unas entidades raras llamadas '*sistemas operativos*' y hasta tal vez, conocer algo de eso llamado programación.

Para colmo se debe mimar para dar vida a un ratón inanimado (*al cual no le gusta el queso*) o tal vez hacer gestos rozando con los dedos pantallas táctiles y entender las reacciones de todo eso, mediante un sinnúmero de símbolos, imágenes e íconos danzando con atractivos colores y sonidos, al compás de la tecnología que las mueve.

Estas pantallas y sus métodos de interacción cambian incansablemente con cada generación, con mucha mayor velocidad de la que se pueden contabilizar o aprender.

Complejidad Innecesaria

Conforme avanza todo, esta fiesta de interacciones, se torna lamentablemente más compleja para hacer actividades o tareas extremadamente simples: como escribir una carta y enviarla a un amigo, guardar y/o imprimir una foto, hacer que una agenda o su celular le avise que debe ir al médico tal día y dónde queda.

Subyace un tema no menos importante y apenas solucionado, que es la manera de lograr que alguien se relacione con sistemas de complejidad creciente, tendiendo a ser hoy casi híper-complejos y de cuestionable usabilidad,

Esto se puede ver con las rampas de aprendizaje que son cada vez más empinadas, hay una necesidad imperiosa de volver a lo simple, contraria a la dirección de la presión de la tecnología que debe justificar el porqué de procesadores y hardware cada vez más rápidos, con requisitos de memoria crecientes, vendiendo '*show-business'*.

Un Sesgo Inherente

Lamentablemente, las personas o profesionales, quienes crean y/o diseñan estos sistemas de ingreso de datos, son muchas veces los mismos desarrolladores o empresas creadoras de la tecnología que subyace. Ellos suelen estar cegados y/o contaminados con tanta cosa abstracta, por lo que terminan creando casi 'engendros' a la medida de ellos mismos y no de la gente común, gente de edad avanzada o niños; quienes ante estas 'creaciones-tecno' tienen sensaciones encontradas entre el miedo y la curiosidad al encender una computadora y no saber que hacer frente a ésta.

Estimación de una Mejora Necesaria

Este trabajo pretende contribuir en el terreno de mejorar y facilitar la interfase hombre máquina en lo que respecta a ingreso de texto, que es un eslabón débil.
Este eslabón, es muy importante pues es en donde la máquina recibe órdenes de qué cosa hacer por parte de una persona, quien en definitiva será el beneficiado por la tarea requerida. Si hay errores, el resultado es incierto, pero por cierto no será el deseado.

Estos mecanismos de interacción "inteligente" hasta ahora descansan en la habilidad de la inteligencia del humano mismo, quien debió "*aprenderse de memoria"* una serie de cosas de "computación", que solamente servirán para lograr que haga lo que desea.

Estimo que la tecnología ya está lo suficientemente madura como para poder plantear que este esfuerzo se reduzca, brindando soluciones mas 'humanas' en lugar de crear escollos en la relación hombre-máquina.

Inteligencia Artificial vs. Real

A lo que se llama hoy *Inteligencia Artificial*, es un término desmedido para lo poco que hoy poseen de *"inteligencia real"* todos estos complicadísimos sistemas como las computadoras hogareñas, portátiles, smartphones, etc., frente a la inmensa inteligencia biológica de una ameba o, sin ir mas lejos, la enorme complejidad de las funciones de una proteína que interviene en procesos biológicos involucrados en la vida misma.

Creo que los mecanismos actuales, si bien han dado resultados importantes y lograron muchas cosas, hay un espacio aún sin resolver y es el cómo hacer que una máquina "*entienda*" realmente y pueda ejecutar una orden verbal, no pre-establecida o memorizada, e interactúe con la gente en un modo más simple y natural, resolviendo cosas en la forma tecnológica más apropiada pero presentándoselas a un humano de manera simple, entendible y a su medida.

Tal vez en un futuro próximo, aparezca esta vez funcionando de verdad, lo mostrado en el pasado de ficción; en donde una computadora acepte una simple línea de texto escrita u oral, con una inteligencia tal que entenderá claramente lo que se le escriba o diga y tan solamente en ciertos casos, nos pregunte algo más o nos pida otro dato.

Procesamiento de Lenguaje Natural

Las ciencias que sin duda arribarán pronto a éstas y otras nuevas metas de relación hombre-máquina, los cuales son los terrenos de la hoy en día *"mal"* llamada Inteligencia Artificial *(IA)* son el nuevo campo llamado *"Procesamiento de Texto Natural" (NLP: Natural Language Processing)* y tal vez se anexen numerosas ramas como una recientemente creada llamada computación con palabras *(CW: Computing with Words)* u otros derivados.

Texto, pero Bien Escrito

No importando cuál es el método de entrada, si deletreamos o hablamos, el sistema elegido para la comunicación de la inteligencia humana, sigue siendo el texto, y si ese texto no está bien escrito, la comunicación falla y por ende los sistemas pasan a ser problemáticos y, en ciertos casos, hasta contraproducentes.

Corrección Natural de Errores

Los humanos ejercemos desde siempre y naturalmente la capacidad de leer correctamente textos en mal estado, escuchar bien las palabras entrecortadas, faltantes y hasta mezcladas con ruidos de toda índole. Si vamos en el futuro, a interactuar más inteligentemente con las máquinas, es hora que ellas aprendan las cosas que nosotros hacemos naturalmente: ***corregir errores***.

3.1 Estado del Arte

El ingreso de texto en los sistemas informáticos, ha resultado una de las principales actividades masivas de las últimas décadas; impulsada recientemente por la inmensa conectividad masiva y la creciente interacción de los usuarios en internet con la web 2.0, foros, blogs, redes sociales, el chat y los mensajes cortos de los teléfonos móviles y en cada vez más electrónica de consumo como las smart-TV, que permiten ingreso de texto.

Razones de Peso

La constante miniaturización de la electrónica, acompañada por su baja de costos y el aumento paulatino de su potencia de cómputo (*ley de Moore*) permitieron a la industria crear nuevos dispositivos electrónicos cada vez más inteligentes para toda clase de usos.

El sinnúmero de nuevas funciones de estos dispositivos requiere del ingreso de órdenes cada vez más complejas, muchas de las cuales se realizan en forma mecánica, usando sensores especializados (*mouse/trackball/tacto/luz*), mientras que el habla directa con conversión voz a texto aún no es lo suficientemente robusto para ser usado masivamente.

Queda por último el teclado alfanumérico como el elemento contemporáneo más usual y robusto para el ingreso de datos y órdenes precisas en forma de texto.

La difusión de estas nuevas tecnologías y el grado de compromiso que presentan por su reducido tamaño, hacen que el ingreso de texto en estos dispositivos resulte cada vez más engorroso y problemático, multiplicando el número de errores que aparecen asociados. Por esto se requiere de nuevas estrategias eficientes y métodos embebibles en su electrónica para abordar los complejos problemas derivados de los errores de escritura. Esto se hace más destacado cuando se procesa lenguaje natural, pues torna potencialmente problemáticos los errores de escritura, como un ejemplo simple: los comandos de sistemas operativos y lenguajes de computación tradicionales, no toleran el más mínimo error.

Celulares y la Predicción de Texto

Un sistema de ingreso de datos para texto debiera de ser tolerante e inteligente, en especial si está orientado a lenguaje natural. Demostrado está el éxito y difusión de técnicas predictivas como el T9, usadas para el ingreso de texto en teclados numéricos de celulares, controles remotos, televisores inteligentes, reproductores y cámaras de foto.

Antecedentes de la Necesidad

Por muchos motivos que se irán analizando, la problemática a resolver para corregir errores de texto es de complejidad llamada ***NP dura***[3], lo cual plantea un desafío, agravado por la restricción de recursos aplicables, impuesta en este trabajo.

[3] Ver problema ***NP-duro*** en el glosario al final del trabajo.

En 1949 *Shannon* [31] publicó su tratado de modelos de comunicación de información como canales ruidosos, aplicable claramente al texto en los libros y numerosos autores como *Kernighan* [60] han usado estas teorías para detectar y corregir los errores.

En el año 1964 *Demerau* [47] ya planteó técnicas para detectar y corregir errores de escritura, en los años 1983 y 1984 numerosos autores, entre ellos *Angell* [48] y *Daelemans* [49]; presentaron técnicas estadísticas atacando la misma problemática, la mayoría sobre idiomas como el inglés y el alemán.

En el año 1984 *Pollock y Zamora* [73], presentan un extenso informe usando el sistema SPEEDCOP sobre las estadísticas de los errores y sus clases en texto inglés.

Otros autores que han hecho a su vez muy buenos resúmenes sacando una foto del estado del arte del problema de ortografía por computadora conjuntamente con las técnicas de corrección y detección en cada momento de la historia; son los trabajos de *Kukich, K.* [56] en el año 1990 y anteriormente, en 1980 *Peterson* [46].

Autores citados, como *Karen Kukich* [56], en los años 1990, luego del analizar las diversas técnicas publicadas, denotaron claramente que había necesidad de mejores métricas realizando un exhaustivo estudio de los métodos de distancia de edición y demás que había hasta ese entonces, clasificando los errores en tres tipos: no-palabras, palabras aisladas y palabras en contexto.

La misma autora luego en 1992 publicó un artículo muy interesante y completo en la ACM *Alberga* [57] que explica y recaba el estado del arte conjuntamente con la efectividad de las diversas aproximaciones al problema hasta ese entonces con las limitaciones existentes junto a las soluciones creadas.

Son numerosas las publicaciones, [60] habiendo diversos enfoques del tratamiento de errores de texto como canal ruidoso. Hay algunos trabajos de *Brill* [63] del año 2000 que se basa en este enfoque para lograr detectar errores y luego tratar de reparar la ortografía.

Al igual que trabajos publicados, también se registraron innumerables patentes en Estados Unidos como las US565771/1995 y las US5907839/1996 entre muchas otras, reivindicando algoritmos y métodos para la corrección de ortografía automática, la mayoría basados en estadísticas con un mix de complejos procesamientos de corpus con altas dosis de heurística y complejas metodologías.

En el año 2000 *Dan Jurafsky* [51] también hizo una clara y concisa reseña del estado de este arte, en el Cap 5 de su libro.

La hipótesis de la necesidad de corrección de texto se refuerza en el 2003 con los trabajos de *Armenta* [30]; donde pone el foco en la interfase texto-voz.

A pesar de todos los desarrollos y patentes creados, *David Yarowsky* en sus trabajos de lingüística [13] del año 2011, aún plantea la necesidad de reconstrucción ortográfica automática. Siendo aún hoy un tema no resuelto en forma satisfactoria.

Implementaciones Comerciales

El primer procesador de texto comercial muy difundido, fue el *WordStar* para *CP/M* el cual incluía un sistema muy básico de revisión ortográfica. Le sucedieron muchos otros como el *WordPerfect* entre tantos que saltaron a efímeras famas, cada uno en su época.

El gigante del software de los 90, *Microsoft* sin dudas embebió en su procesador de textos *MS-Word* un corrector ortográfico propietario desde el principio, del cual no hay especificación, mediciones de calidad ni documentación disponible, salvo promesas. Si bien este procesador de texto *MS-Word* es bastante bueno, no ha sobrevivido mucha competencia con que comparar; habiendo claros indicios de sus flaquezas: por lo general las correcciones automáticas terminan desactivadas en casi todas las instalaciones, siendo una conocida tortura para los usuarios, por su mal funcionamiento.

En el 2004 *Martín Reynaert* [14] desarrolla un sistema corrector para inglés y francés; logrando cifras de mérito interesantes basadas en corrección bajo contexto, comparando contra el *MS-Word* y el *ISpell y sus derivados*. En este trabajo la novedad consiste en usar estadísticas de bigramas de palabras y una función de dispersión especial para determinar rápidamente si una palabra pertenece a un diccionario de palabras existentes que ayuda para hallar un candidato en caso de no existir. Esto según entendemos, en castellano es mucho más complejo de implementar.

En 2007, *Peter Norvig* [52] describió de nuevo en forma clara y simple como construir un corrector ortográfico básico, dando ejemplos de esta aplicación en lenguaje ***pyhton***

Hay un trabajo reciente del 2010 de *R. Mitton* [85] que evalúa el estado del arte en ese momento y analiza los diferentes métodos utilizados desde el principio del problema mismo, citando un gran número de trabajos y autores que datan desde 1962.

Considerando el estado del arte actual, en el año 2013, en lengua española y su procesamiento automatizado; los productos y bibliotecas más importantes del mundo en esa lengua, aún no poseen incorporados correctores ortográficos satisfactorios pero tampoco hay consensuada una métrica clara ni estándares para seguir.

Ambientes de Desarrollo y Código Abierto

Uno de estos productos para el tratamiento de texto en lenguaje natural, es el creado por los investigadores del *Politécnico de Cataluña*, en España (*UPC*) llamado *Freeling* [15] que contiene un analizador morfológico que usa muchos recursos computacionales, siendo el único de desarrollo abierto, completo y aplicable al español. Hasta mediados del año 2013, su versión 3.0 aún no poseía módulo alguno de corrección de errores de ortografía y lo más parecido es realizar un etiquetado de palabras desconocidas, basado en sufijos máximos, el cual falla cuando un error de ortografía está justamente en el sufijo y en todos los casos, no corrige nada.

Hay otros pocos ambientes de desarrollo basados en código abierto, como desde el año 1995 el proyecto GATE [4]de la *Universidad de Sheffield* [10], UK el cual posee

[4] Proyecto GATE http://www.gate.ac.uk de Universidad de Sheffield, UK

numerosísimos módulos y plug-ins, pero la parte de corrección ortográfica es pobre y no posee recursos que se haya podido usar libremente para el español.

No hemos hallado proyecto de código abierto salvo los mencionados en esta tesis: *ASpell* [7]*, ISpell, Hunspell, MySpell, OpenOffice, LibreOffice* y sus derivados que hayan atacado efectivamente el problema de errores de ortografía.

Corrección de Código Libre

Contados proyectos iniciales, desde los años 90, devinieron en productos que han trascendido hasta estos días, son software libre y uno emblemático se llama ***MySpell*** [5]

Basándose en este proyecto, *László Németh* en Hungría y con el auspicio de la "*Budapest University Media Research Centre*" (*BME-MOKK*) creó un producto junto a una librería **C++,** llamados: ***Hunspell*** [6] la cual se usa ampliamente en *OpenOffice*. Otros correctores ortográficos como el GNU-***ASpell*** [7], derivaron de ellos y poseen módulos agregables como "plug-in" para numerosos procesadores y navegadores.

Estos correctores son usados en procesadores de texto de amplia difusión como el ***OpenOffice***, ***Libre Office***. También se usan como *plug-ins,* para navegadores como el ***FireFox*** y el ***google-Chrome*** y muchos otros del mundo de software libre.

Todos ellos se destacan por ser de uso libre pero de mediana calidad; basándose la mayoría de ellos en compresión morfológica de afijos sumada a una búsqueda simple de errores restringidos a pocas letras, con algunos aditivos fonéticos en los más avanzados.

Otro aspecto a considerar es que la calidad va acorde al recurso lingüístico y el costo de crear su diccionario asociado y mantenerlo es muy alto. Por eso se hacen productos compatibles con los diccionarios morfológicos originados en el proyecto ***MySpell***, enriquecidos comunitariamente a lo largo de décadas.

La conclusión es que casi todos estos productos poseen cifras de mérito similares, las cuales son fuertemente dependientes del diccionario de base, que todos comparten.

Restauración de Acentos y Diacritos

Dentro de todos los posibles errores de ortografía, los más frecuentes en idiomas con marcas diacríticas *(acentos, diéresis, etc.)* son justamente la omisión o confusión de esas marcas. Diversos autores afirman que la prevalencia de este tipo de errores se sitúa entre el 80 y 90% de todos los tipos de errores. Esto, ciertamente, ha generado una creciente preocupación por este tipo de errores, buscando métodos para su detección y posible corrección en forma automática.

En numerosos trabajos se describen infinidad de técnicas específicas para *restaurar acentos* y si bien esto es importante, es tan solo una parte del problema de los errores en

---

[5] MySpell http://en.wikipedia.org/wiki/MySpell
[6] Hunspell http://hunspell.sourceforge.net/
[7] ASpell http://en.wikipedia.org/wiki/GNU_Aspell

los textos. Hay un trabajo del año 1994 [81] que utiliza listas de decisión para resolver esto para español y francés. Uno de los últimos trabajos muy interesantes en el tema es el publicado en 2012 por *Atserias & all.* [55] en el cual se enfoca en el español, además de resumir y citar una gran cantidad de fuentes útiles; analiza y explica el estado del arte del tema; presentando y comparando luego las cifras de mérito de diversos métodos, incluyendo los productos comerciales más comunes como ***Word2007*** y algunos on-line como ***Google Docs*** y ***correctorortografico.com***

Las evaluaciones citadas [55], arrancaban en cifras de mérito o calidad de restauración de acentos, usando una medida estadística llamada ***F*** del 82% en métodos que usan técnicas estadísticas similares a las que se usaron en esta tesis, contra apenas 30% medidos en el *MS-Word 2007*, ¡cayendo a menos del 1% para los servicios online!

Resumen

Si se tiene la palabra correcta en el diccionario del corrector y éste itera lo suficiente permutando letras, tarde o temprano arribará a la palabra correcta; el asunto que pesa es cuánto se tarda y si por hacerlo con el método que sea, se pasa primero por muchas palabras no-correctas, generando una lista tanto larga como inútil. O sea si se usan muchos recursos de memoria, disco y procesador, se puede corregir cualquier cosa. El desafío consiste en hacerlo ***eficazmente*** (*pocas opciones + la primera la correcta*) y ***eficientemente*** (*rápido+ poco uso de procesador y memoria*) lo demás de cómo está hecho el diccionario, si es ampliable o si el software puede correr en múltiples plataformas o no, son temas comerciales o de implementación; el algoritmo subyacente es lo primordial y mandará por sobre todas las cosas.

3.2 Tendencias

Google: *Ud. tal vez querrá decir..?*

Se podría afirmar que esta tesis está alineada con los más recientes avances en el tema: Se observa que Google, hace uso de inteligencia artificial, colectiva y lingüística predictiva para ‘adivinar’ y ‘sugerir’ texto cuando estima que está mal escrito. También el gigante informático Microsoft, en su producto estrella: *Windows 7* incluyó una línea de búsqueda textual inteligente en su menú.

Surgirá una mayor necesidad de estas técnicas cuando los sistemas puedan dialogar con el usuario en lenguaje natural, habiendo numerosas evidencias de esta tendencia.

Apple Siri

Recientemente en el 2010 salió al mercado un sistema llamado ***Siri*** para los productos de *Apple* como el ***iPhone, iPod e iPad*** el cual ofrece interacción conversacional con algunas aplicaciones, como los recordatorios, el tiempo, la bolsa, la mensajería, el e-mail, el calendario, los contactos, las notas, la música y los relojes.

Es un asistente con reconocimiento y respuesta por voz, útil para responder cosas sencillas y realizar algunas operaciones con los dispositivos. Está basado principalmente en procesamiento remoto de habla y diálogo en lenguaje natural. En ***Siri*** la voz es

digitalizada en el celular, luego se envía a los servidores centrales de Apple por la red móvil, quienes realizan el reconocimiento de habla y la lógica de diálogo para la determinación de la respuesta, enviando al celular del usuario la respuesta en forma de voz sintetizada junto a las órdenes asociadas.

Este esquema de teleprocesamiento, claramente se fundamenta en la escasa capacidad de proceso y almacenamiento disponibles en los dispositivos móviles. A pesar de ser impresionantemente grande, aún no alcanza para el procesamiento de habla, junto a los complejos sistemas de diálogo, necesarios en lenguaje natural.

Métricas y Corpus

Se estima que dada la dificultad de comparar métodos y técnicas de corrección, aún hacen falta métricas y estándares claros y corpus[8] anotados en numerosos idiomas, para lograr una medición de la calidad, bondad y el adecuado contraste entre todos los sistemas de corrección existentes y los por venir.

Problemas No Resueltos

De hecho el poder determinar si un apellido o nombre propio está mal escrito en una base de datos, es un tema abierto y los sistemas actuales rara vez revisan esos campos de texto por no disponer de herramientas ni algoritmos para esto; dejándolos tal cual se ingresaron, probablemente llenos de errores, que pasan sin ser detectados, generando un sinnúmero de problemas cuando esos datos son usados posteriormente. Lo estricto en las comparaciones de los campos de texto, trae importantes consecuencias debido a que por no corregir ni detectar errores, se toma como válido lo ingresado, llevando a resultados erróneos. Esto se agrava en ámbitos sensibles como los son los temas legales y administrativos, cruzado de información, minería de datos, operación bancaria, crediticia, o fiscal, entre muchas otras situaciones.

Reflexión

Curiosamente, si nos ponemos a pensar en el tema de datos textuales, casi cada texto único existente en un ordenador, fue originalmente ingresado por una persona.

Pero las palabras originales junto a su deletreo y secuencia, al igual que las órdenes, incluyendo toda la programación; tienen su indiscutible origen directo o indirecto, en el intelecto humano y el accionar de teclados.

Esto trae como corolario el inmenso número total de errores existentes y presentes en la información contenida en casi todos los sistemas informáticos; en donde la mayoría de los ingresos de datos, reiteramos: fueron hechos por humanos.

---

[8] ***corpus***: conjunto de recursos lingüísticos con ciertas características. Suelen ser extensas selecciones de texto en un idioma en particular, obtenidos de diarios, literatura, notas periodísticas, correos electrónicos, blogs, publicaciones científicas, etc. Su uso es principalmente para disponer de una muestra significativa de texto natural, por lo general de numerosos millones de palabras, basados temática e idiomáticamente en el set de origen y su idioma, suelen poseer estadísticas y etiquetado o anotaciones internas asociadas.

Tan solo el número de letras y caracteres nuevos ingresados hoy en día, segundo a segundo en todo el planeta, en los teclados de los miles de millones de computadoras y dispositivos móviles o portátiles, es una cifra que desafía el asombro. Lo que también es preocupante, es que el número de errores de texto es proporcional a esta cifra y aunque sea escasa la proporción, el número total es muy grande.

3.3 Motivación

Si alguno de estos métodos usados por sistemas comerciales, los enunciados en patentes, junto a todos aquellos creados con anterioridad citados en trabajos científicos, hubiesen sido realmente exitosos y el problema se hubiera resuelto; éstos métodos estarían entre nosotros, presentes en todos los dispositivos, solucionando casi todos los errores y la motivación de esta tesis no existiría, pero no es así según numerosos autores [56][85].

Si tan solo se logra con este trabajo, mejorar y mitigar el problema de los errores de ingreso del texto en dispositivos electrónicos, a bajo costo computacional y con aceptable calidad, seguramente se habrá aportado algo valioso al estado del arte.

Futuro Próximo

Según la visión de este autor, el creciente número de textos ingresado en computadoras y en dispositivos móviles, tanto sea mediante SMS, chat, MSN-messenger, skype, whatsup, twitter, blogs como facebook, blogger, etc.; además de algunos sistemas incipientes de diálogo como ***Siri*** de Apple marcan juntos una clara tendencia de comunicación en lenguaje natural, mostrando las necesidades y gustos emergentes de la sociedad.

La cantidad del texto total generada periódicamente por todas estas tecnologías supera fácilmente la suma de toda la literatura existente en las bibliotecas del planeta. Sin duda ese texto creado segundo a segundo, no suele estar revisado ortográficamente y seguramente contendrá una inmensa cantidad de errores, generando numerosos inconvenientes.

Simultáneamente en el mercado están apareciendo un grupo creciente de soluciones basadas en procesamiento inteligente del habla y otras usando diálogo en lenguaje natural.

Las redes sociales, la web 2.0 y sus derivadas, pueblan la web de información en forma de texto mayoritariamente. Esta información es muy rica para realizar un profundo análisis obteniendo estadísticas sobre opiniones, productos, tendencias, temas de discusión, etc. Esto se realiza mediante minería de textos y procesamiento de lenguaje, el cual en el caso de contener errores corrompen los resultados. Por eso es imperiosa la necesidad de un mecanismo automatizado para la reconstrucción de errores.

Todo esto refuerza y fundamenta aún más la necesidad de nuevos desarrollos de calidad, con aplicación de todas las herramientas de la ingeniería en el área, enfocados a necesidades locales no satisfechas como el tratamiento de textos con errores, en el idioma español y otros lenguajes flexivos, de reconocida problemática.

4 Análisis del Problema

En esta sección se realizará un análisis del fenómeno de texto en lenguaje natural, al amparo de las teorías de comunicaciones, al tiempo que estudiaremos los mecanismos involucrados en su generación, codificación, transmisión y posterior decodificación.

Una vez realizado esto se individualizarán los tipos de segmentos a detectar, utilizando un método de segmentación y clasificación inicial, para luego realizar una clasificación mas fina en una segunda etapa, conforme a las necesidades de reconocimiento clásicas para diferenciar los segmentos gramaticales de los agramaticales y luego descubrir entre ellos los formatos de los números, fechas y horas, unidades, puntuaciones, fórmulas químicas, entre otras tantas cosas factibles de ser escritas con texto en lenguaje natural.

Ni bien estén individualizadas y clasificadas las partes, lo que sigue normalmente es que se entregue toda la secuencia a un bloque posterior, el cual realizará una sofisticada tarea para lograr el reconocimiento de segmentos, llamada **NER** (*Named Entity Recognition*), en esta etapa se detectan y unen bajo una misma etiqueta o clasificación: nombres de compuestos, números expresados como palabras, abreviaturas y locuciones, entre otros muchos segmentos aglutinables.

Esto último se menciona, ya que también es posible que el mecanismo de detección de segmentos a analizar falle debido a errores de diseño del método, como por errores y/o ambigüedades propias del texto. Al detectar estas cosas, se deben realizar tareas recursivas, es decir reanalizar subsegmentos de lo analizado, o todo lo contrario: unir segmentos para luego reanalizarlos.

4.1 El Lenguaje como secuencia de símbolos

Hay una enorme base de teorías sobre lenguajes formales. Algunas de las más interesantes, claramente planteadas y reglamentadas, que han aportado un sinnúmero de herramientas, son las teorías de generación de lenguajes basados en *Noam Chomsky*[9].

Según *Chomsky* [8] la estructura básica del lenguaje y los patrones básicos residen internamente y son innatos a los seres humanos. Esta teoría se denomina gramática universal y esboza que esencialmente, los seres humanos, al nacer, tienen un conjunto de reglas integradas en ellos. Estas reglas permitirían a los seres humanos la capacidad de aprender cualquier lenguaje a través de la interacción. Un fuerte elemento de prueba de esta teoría es el hecho de que los niños, a través de un corto período de tiempo, adquieren la capacidad de producir y entender un número elevado de oraciones.
Sin embargo, modelos adecuados para recrear esta plasticidad no han sido aún creados.

[9] Noam Chomsky “The Noam Chomsky Website”, http://www.chomsky.info/

Gramáticas

Estas teorías definen y clasifican lo que es un lenguaje, según cómo sean sus reglas de generación, llamadas ***gramáticas generativas***, permitiendo su aplicación a la lingüística; prediciendo su comportamiento y logrando ricas clasificaciones basadas en la teoría planteada.

Con esto se facilita la creación de algoritmos que resuelven si una secuencia de símbolos pertenece o no a una determinada gramática con ciertas características de regularidad. Esto último se llama ***parseo*** (palabra inventada y derivada del inglés: *parsing*). El ***parsing*** y los métodos asociados, son tan útiles y necesarios en informática que constituyen el ABC de todo lenguaje de computación creado o por crearse.

En cambio, en la teoría de comunicaciones y de la información, el tema principal pasa por la medición de la cantidad de información, sus estadísticas, los mecanismos de codificación, transcripción, la aparición de errores y todas las fórmulas asociadas.

Se tratará de explicar la asociación entre una secuencia de texto con un conjunto de mensajes de información bajo el criterio de transmisión de la información clásica y su codificación.

Sobre Signos y Letras

Nuestro lenguaje natural es simbólico y de hecho nosotros utilizamos un conjunto finito de signos gráficos o grafemas para armar las palabras escritas que conocemos, esos signos los llamamos letras del alfabeto y en ***español*** su número es de apenas 28, desde la **a** hasta la **z,** sin contar la *'ll'* y la *'ch'* como letras.

Ahora, si diferenciamos las mayúsculas de las minúsculas, entonces se duplican y su número se eleva a 56, si además incluimos como signos las letras acentuadas, las con diéresis y con 'crema' (*macron*) como la **ñ**, pues debemos agregar otras 14 más. Pero resulta que también tenemos alrededor de 10 números, además de signos de puntuación, los cuales muchas veces son usados en forma variada conforme al lenguaje o la cultura regional.

El resultado final es conocido como 'set de caracteres' y se han creado diversas "tablas" de signos, asignando un número único a cada uno. Por ejemplo, para el inglés se ha creado inicialmente una norma ANSI, conocida como el alfabeto ASCII con 127 signos, el cual luego se extendió a 256 por comodidad dado el uso frecuente de 8 bits (*unidades binarias*) que cómodamente representan 256 números diferentes.

Finalmente se han extendido para su aplicación a numerosos lenguajes indoeuropeos, dando origen a la norma ***ISO 65000***, llamada ***Unicode***, la cual define un alfabeto universal, el cual incluye un enorme número de símbolos y prevé hasta "*espacios de códigos*" para posibles lenguajes extraterrestres, por extraño que parezca.

El ***Unicode***, de alguna manera posee una enorme cantidad de símbolos y letras, de las cuales frecuentemente se usan una pequeña parte. Conforme al contexto cultural y de lenguaje en español tradicional, incluyendo algunos "*diacritos*", el número de símbolos

suficientes para expresar escritos normales, es menor a 256, incluyendo símbolos para espacios, dígitos, puntuaciones clásicas, tabuladores, marcas de párrafo, etc.

Espacio de Secuencia

Cuando se tiene una secuencia de un número finito de símbolos y se los pretende analizar, se recurre a un espacio de posibles combinaciones diferentes, en el caso de un lenguaje de un idioma indoeuropeo, utilizando alfabetos latinos, éste resulta acotado.

Éste espacio de variabilidad, se reduce a alrededor de 30 a 40 símbolos (*letras*) diferentes para construir las palabras, contándolas como una secuencia de letras, terminada por un espacio o un signo de puntuación. Veamos un ejemplo:

> *El carro del **verdulero** venía tirado por caballos, uno de ellos estaba enfermo.*

En este caso cada segmento separado por espacios se podría considerar una 'palabra' del mensaje, ahora nos preguntamos que diferencia este segmento de este otro:

> *Lio dregjluis jlioHjg, **Dilatskn56jk** jad68 fsdfl72d2, dgknidj259sg 254 moAhslk.*

Segmentos diferenciables o mensajes

Cuando un mensaje se codifica, por lo general se definen segmentos con formatos, tal es el caso del lenguaje natural en donde estos segmentos son de longitud variable y algunos de ellos se llaman palabras, otros símbolos, números, fórmulas y muchos más. Los separadores de estos símbolos, suelen ser signos de puntuación, de lista, de párrafo, retorno de carro, tabuladores, espacios, entre otros

---

4.2 Las Palabras

Resulta que una determinada 'palabra' de nuestro lenguaje es un segmento concreto de texto, del ejemplo anterior: ***verdulero*** o en el segundo caso: ***Dilatskn56jk*** los cuales debiesen figurar junto a su significado, en algún sitio accesible, que contenga información útil asociada al segmento, comúnmente llamado ***diccionario***.

Las palabras como macro-símbolos

Como las palabras deben estar acuñadas en esos recursos lingüísticos llamados ***diccionarios***, su número se presume finito. Esto nos permite analizar la variabilidad de las mismas usando simple matemática combinatoria.

Veamos pues, un sencillo análisis para el español. Si eliminamos la distinción entre mayúsculas y minúsculas, podemos asumir que hay 28 (a-z) + 7 (áéíóúüñ) = 35 letras diferentes, con lo cual una palabra de N letras pues podría escribirse de 35^N maneras diferentes. El número total de palabras posibles crece exponencialmente con el número de letras, lo cual usaremos para su análisis en las siguientes secciones.

4.3 Constituyentes del Lenguaje Natural

Cuando se escribe texto, debido a la tecnología existente, éste puede contener una gran variedad de elementos desde simbólicos específicos, puntuaciones, hasta puramente gramaticales, además de las clásicas palabras, éstas partículas o subconjuntos de símbolos, números y letras, son frecuentemente difícilmente distinguibles por un sistema automatizado y presentan serios problemas de ambigüedad, necesarios de resolver en cada caso en particular.

Para no alargar esta sección, se definirán 2 macro-clases de segmentos de letras o símbolos reconocibles como símbolos en el lenguaje humano: las ***palabras*** ortográficas y los ***macro-símbolos*** que tienen otras codificaciones.

Vamos a explicar esto brevemente, si llega al analizador un texto ingresado como éste:

Hoy @jorge34 twitteó a las 2:15pm un msj. del 10% para graciela@gail.com!

Este texto además de las palabras contiene partículas que deben ser etiquetadas como lo que es cada parte: una dirección de e-mail (*graciela@gail.com*), una dirección de de twitter (*@jorge34*), una presunta hora (*2:15pm*) y una abreviatura (*msj.*), un porcentaje (*10%*) junto a las demás palabras de diccionario. Esto es necesario para 'decirle' a las etapas posteriores de proceso, qué son, para saber qué hacer con ellas y como tratarlas.

Necesidad de Corrección de Errores

Se estima que no tiene ningún sentido corregir ortográfica ni tipográficamente a los elementos no-presumiblemente gramaticales y menos aún buscar reglas o métodos para esto, dado que son reglas rígidas como la escritura de un número, si falta se agrega, o se permuta un dígito, es simplemente otro número; no hay manera sobre la tierra de contrastarlo, del mismo modo que si falta el exponente, el signo o el punto decimal.

Escapa a este trabajo intentar corregir en una fórmula matemática si falta una variable, sobra un operador o en una fórmula química hay cambiado un componente, falta un paréntesis, o en un código de alfanumérico de un componente tecnológico falta o sobra una letra, guión, símbolo etc. El correcto tratamiento de estas entidades será tarea de una etapa posterior o no. Todo esto lo consideramos fuera del contexto de este trabajo.

En el **apéndice 5** se detallan todos los tipos de elementos o entidades consideradas en el presente trabajo, con una explicación detallada de cada uno, ilustrado con ejemplos.

En el próximo capítulo se explica cómo se logra segmentar una secuencia de letras y caracteres para identificar esos segmentos llamados palabras, los símbolos, las puntuaciones, entre otros muchos más.

Se prosigue con el análisis que alimenta la inquietud de este trabajo, que son las palabras gramaticales y sus errores asociados al ingreso de las mismas por un humano.

4.4 Espacio Vectorial de Palabras

Las palabras formadas por una secuencia de letras, conforman símbolos únicos y distinguibles entre sí. Analizaremos el número posible de estas combinaciones únicas y veremos cómo se puede considerar estos conjuntos según diversos criterios.

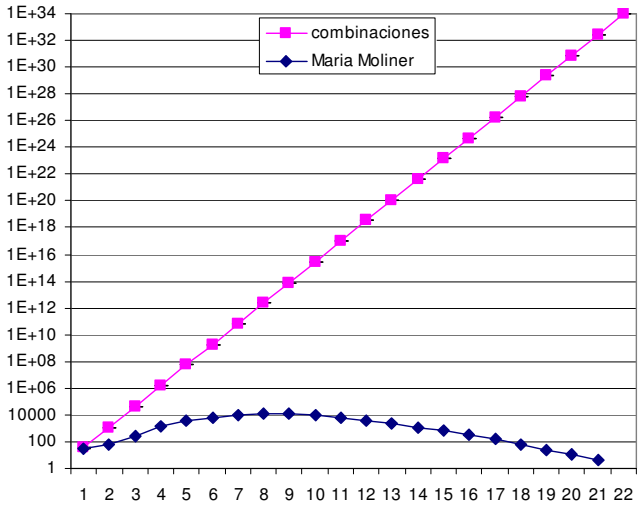
Inicialmente se infiere que este espacio de símbolos llamados 'palabras' es muy grande a partir de cierto número pequeño de letras, si en español hay palabras de hasta 22 letras[10], calcularemos en detalle estas dimensiones de variabilidad.

Este espacio suele usarse en forma vectorial, ya que se podría asignar cada palabra como un punto de un espacio multidimensional y sus ubicaciones relativas (*coordenadas*) estar definidas conforme las características que se deseen conseguir o representar, por ejemplo palabras similares en pronunciación podrían tener una representación dada, en la cual se pueda definir una función de distancia y formar grupos o 'clusters', como ejemplo.

Todo esto y mucho más se conoce bajo el término de *Vector Space Model* (***VSM***) y hay abundante literatura [70] sobre el tema, dependiendo del enfoque.

En el siguiente experimento Fig 1. se graficó el número de palabras diferentes de N-Letras del diccionario español de María Moliner, con 68 mil palabras.

Fig. 1



Número total de posibles palabras en un diccionario vs. las combinaciones de letras, en modo logarítmico para poder contener en forma visible toda la información, por la escala exponencial.

En el gráfico, se observa que a medida que el número de posibles combinaciones crece, el número total de palabras existentes de esa longitud mantiene una relación de exponentes razonable entre las 1 y las 6 letras, luego se mantiene constante y finalmente decrece suavemente a partir de las 8 letras, llegando a 21 letras como máximo.

[10] En realidad en español hay palabras de hasta 46 letras, pero éstas son excepcionales y se presentan bajo características de palabras científicas originadas por concatenación/fusión de otras palabras.

Esto nos enseña algo importante: el espacio de combinaciones posibles de letras es muchísimo más grande que el de las combinaciones utilizadas o aceptadas, creciendo esta diferencia exponencialmente a partir de las 6 letras; lo cual permite asegurar que hay un enorme espacio estadístico, suficiente para corregir errores dentro de una secuencia cuyas palabras deben estar debajo de la curva azul (recta), y por errores se hallan en el otro lado.

Imposibilidad de Hallazgo Directo

Lamentablemente, también nos enseña que no toda combinación de letras resulta en una palabra existente, y esto es tanto más grave cuanto mayor es el número de letras a partir de las 6 letras de longitud. Veámoslo en un cálculo de aproximación sencillo.

Por ejemplo, en este diccionario del español, hay 9307 palabras posibles de 10 letras y $2{,}75 \times 10^{15}$ combinaciones de letras, apenas 3,37 palabras 'reales-existentes-viables' por cada millón de millones (10^{12}) de combinaciones de letras.

Esto nos da una fuerte pauta de que el poder hallar una palabra con alguna falla en 10 letras, constituye un problema formidable, debiendo ensayar un número sideral de combinaciones. Los problemas de este tipo se llaman NP-duros[11] dado que el espacio de búsqueda es de características combinatorias y por ende no resulta resoluble en tiempos polinomiales (*léase razonables*).

Esto último limita clara y seriamente la posibilidad de realizar búsquedas por tanteo, a partir de palabras halladas con presuntos errores. Un cálculo sencillo arroja la imposibilidad, si una consulta de pertenencia a una base de datos extremadamente veloz de palabras, tarda aprox. 1.0 mS, el tiempo de explorar el espacio de palabras de 10 letras sería de 9,39 años, resultando imposible su utilidad práctica. Para más letras el número tenderá a ser astronómico, evidenciando aún más la imposibilidad del uso de la fuerza bruta.

Este tipo de problemas son de los llamados problemas abiertos, siendo abordados en combinación por la ingeniería, informática, matemática y estadística para plantear alternativas de solución. El aprendizaje automático y la inteligencia artificial proporcionan algunos métodos que solo en contados casos se puede demostrar que son óptimos; en otras palabras si se desea hallar una solución en un universo combinatorio de resultados, se puede arribar a la misma explorando un número significativamente menor del mismo, en un tiempo razonable, usando alguno de estos métodos.

---

[11] Ver problema ***NP-duro*** en el glosario al final del trabajo.

4.5 Hipótesis de Lógica Subyacente

En otras palabras, hay espacio para hallar las palabras adecuadas dado un presunto error, pero el hallazgo de la palabra correcta responde a un problema combinatorio de alto grado, presumiblemente imposible de resolver en tiempos razonables.

De hecho si a un humano se le presentan un conjunto de palabras de diversa longitud, con fallas ortográficas de varios tipos, en lugar de tardar exponencialmente respecto al largo de la palabra, asumiendo un presunto proceso mental, éste reconocerá la palabra correcta en muy breves instantes, casi un 100% de las veces, siendo casi independiente de la longitud, siendo tanto más veloz cuanto más larga es la palabra.

Es muy probable que el humano lo resuelva la mayoría de las veces, mientras lee la palabra por primera vez y no tendrá dudas al respecto, lo cual realmente resulta asombroso.

Este fenómeno se halla evidenciado por numerosos experimentos cognitivos [72], de hecho muy complejos de realizar y medir, pues tratan con percepción humana lo cual no es un número exacto, sino un número discreto de fallas sobre el total de experimentos, y no se puede realizar una cantidad significativa de experimentos pues el tiempo humano está limitado y se tarda un tiempo 'normal' en leer una palabra y decir que entendió.

Es sabido que la lectura humana es un proceso muy robusto, complementado por la comprensión semántica, el conocimiento previo y una fuerte estimación contextual.

Lectura Robusta

De hecho una persona puede leer fácil y rápidamente un texto con un asombroso porcentaje de letras mal escritas, permutadas u omitidas. Para ilustrar lo antedicho daremos un ejemplo simple en donde el contexto construido a medida que se lee, ayuda más aún para reconocer las palabras, aumentando la velocidad a medida que se lee:

Sgeun etsduios raleziaods pro una uivenrsdiad ignlsea, no ipmotra el odren ne el qeu lsa ltears etsen ecsritas, la uicna csoa ipormtnate es qeu la pmrirea y la utlima ltera esetn escritas en la psiocion cocrreta. El retso peuden etsar ttaolmntee mal y aun pordas lerelo sin pobrleams, pquore no lemeos cada ltera en si msima snio cdaa paalbra en un contxetso. Presnoamelnte, etsa csoa me preace icrneilbe!

Si Ud. leyó fácilmente esta párrafo, tanto más velozmente cuanto más avanzaba, entonces es una persona normal y además esto último evidencia que en el lenguaje humano deben existir uno o varios patrones subyacentes con la redundancia suficiente para permitir mecanismos exitosos de reconstrucción de datos, basado en una hipótesis de que probablemente existan ciertas codificaciones desconocidas aprendidas por nuestro cerebro, que tal vez sean del tipo lógico y/o heurístico, pero sin duda son pertenecientes a las palabras, su secuencia de letras asociados a la estructura de cada idioma humano en particular.

En consecuencia, tal vez el hallazgo o modelización de este tipo de lógicas y su aplicación, permitiría reconstruir una palabra correctamente a partir de indicios muy

precisos, similar a como lo hace un humano, con el beneficio de un menor tiempo de proceso.

Esta última es precisamente la hipótesis principal que mueve la dirección de esta tesis.

---

5 Las Comunicaciones

Para poder modelizar bien el mecanismo de los errores en las palabras, es conveniente adentrarse en sus orígenes y analizar los mecanismos tanto de formación como de su análisis, para eso haremos un breve repaso de la evolución asociada a este tema.

Luego se analizará el detalle y funcionamiento completo del canal de comunicación humano que se considera en esta tesis, para luego usar esto para el desarrollo del algoritmo de restauración léxica.

Introducción

La necesidad de las comunicaciones estuvo presente desde siempre en la evolución de las especies; en el caso de la humana, ésta necesidad se abstrajo y debido a la aparición del intelecto y el pensamiento abstracto, comenzó a estar presente mucho antes de la formalización del lenguaje natural, persistidos hasta nuestros días mediante evidencias en petroglifos con pictogramas, dibujos y símbolos, como los de Altamira.

El lenguaje humano y las comunicaciones en consecuencia, nacieron como un todo, luego se formalizaron y diversificaron, dados los medios que se disponían. Se verán ahora en detalle los mecanismos sobre los que se desarrolló el presente trabajo.

5.1 Mecanismos de Comunicación

El modelo de un sistema de comunicaciones en electrónica y telecomunicaciones, por lo general está separado y abstraído de los mecanismos originales de comunicación, esto permite modelizar, estudiar, crear y aplicar algoritmos en las diversas etapas, sin tener en cuenta el mensaje.

Sin embargo no debemos olvidar el origen último de las comunicaciones humanas, pues es de lo que trata este trabajo; y es la comunicación mediante palabras.

Su Prehistoria

Es importante recordar, que los sistemas de comunicación a distancia entre humanos fueron siempre de origen cognitivo y simbólicos, iniciándose probable y naturalmente con gestos de transmisión visual y/o táctil, asistidos tal vez mediante sonidos de variado tipo.

De hecho los animales usan variadísimos mecanismos y emplean lenguajes muy sofisticados, la mayoría de ellos aún hoy resisten el análisis científico.

La versatilidad necesaria para lograr una transmisión a distancia, independiente de factores como la luz, el contacto directo, la posición relativa de las partes y la distancia, probablemente destacaron al sonido como elemento ideal de comunicación.

La necesidad de transmitir sonidos fácilmente distinguibles asociados a conceptos, usando recursos propios y reproducibles en todo contexto, y que trasciendan toda la

especie, fueron factores que seguramente provocaron la sofisticación evolutiva del aparato fonoarticulador humano, logrando ser capaz de generar un puñado de sonidos claramente distinguibles entre sí y de los sonidos comunes de la naturaleza. Habían nacido los fonemas.

Conforme evolucionó la especie humana en sofisticación, se originó un creciente número de conceptos a transmitir. La dificultad de crear una suficiente variedad de sonidos claramente diferentes y reconocibles, llamados fonemas, sin dudas nos obligó a crear estrategias de agrupación. Había surgido la palabra hablada.

A medida que estas palabras se fueron sofisticando para lograr la codificación de ideas de complejidad creciente, fueron surgiendo lógicas de orden superior relacionadas con la posición de estos segmentos en secuencias que codificaban ideas completas, que llamamos sintagmas, originando los lenguajes aglutinantes indoeuropeos.

La búsqueda de la transmisión mediante mecanismos no tan lábiles y limitados en tiempo y espacio como la palabra hablada, seguramente originó diferentes estrategias de perpetración, tal vez mediante símbolos y objetos, pero como no eran reproducibles y fácilmente duplicables, dio origen al lenguaje escrito.

Desde los originales sonidos que tal vez usaron nuestros antepasados y contemplando hoy lo único que nos legaron, representando ideas en forma de petroglifos, y pasando por toda la evolución de la cultura humana, es que se culminó en la palabra escrita actual, usando para representarla, desde ideogramas hasta alfabetos.

Todo esto que evolucionó naturalmente, luego se estudió, apareciendo ciencias como la lingüística, que trató de descubrir las reglas intrínsecas para luego reglamentarlas.

Medios Físicos

Los únicos elementos que nos comunican desde el exterior físico, al cual llamamos realidad, son nuestros limitados sentidos; ellos poseen diversas capacidades de manejo de información, veamos los asociados a las comunicaciones básicas:

sonidos
- ***vocales-guturales,***
- ***realizados mecánicamente mediante instrumentos***

visual
- ***pictogramas y representaciones***
- ***gestual (movimientos abstractos y representativos)***

táctil
- ***abrazos, caricias, golpes, empujones, heridas***

Dado que no se desea hacer un tratado de historia de las comunicaciones ni de la evolución, en este trabajo nos vamos a concentrar en elementos u objetos, que son representables con palabras.

Estos *objetos* que se transmiten aún hoy, de persona a persona son las ***ideas***, codificadas mediante una gramática que provee marcos de normalización: léxico, reglas de sintaxis, ortografía, todas asociadas a un determinado lenguaje. Estas ideas resultan finalmente expresadas mediante secuencias de símbolos, llamadas palabras.

Las palabras son creadas morfológicamente y acuñadas en el léxico, las cuales hoy se transmiten de numerosas maneras diferentes, concentrándonos en las siguientes que resultan las de mayor frecuencia:

Vocal (sonido)

Textual (Visual)

Esta última: la transmisión escrita, se mantuvo intacta por varios milenios, hasta que el ingenioso orfebre alemán *Johannes Gutemberg*, en el siglo 15, inventó un modo de duplicar rápida y económicamente la palabra escrita, creando la imprenta y acelerando su multiplicación a bajo costo y en consecuencia el devenir del saber con un factor importantísimo, esto hizo entrar a la humanidad en una vorágine de información, promoviendo el avance del conocimiento con un factor geométrico en el tiempo.

El Comienzo del Avance

Se podría decir con certeza que *Gutemberg* con la imprenta, cambió definitivamente y para siempre el valor del coeficiente de la multiplicación del conocimiento humano, creando un primer crecimiento netamente exponencial.

La Aceleración

Hoy la informatización junto con ***internet*** dieron una nueva vuelta de tuerca a este coeficiente de multiplicación y difusión del conocimiento humano, tornándolo mucho más explosivo que la imprenta, hace apenas 500 años.

Una Reflexión

Pero como todo es consecuencia de algo anterior, si los hechos no hubiesen seguido su curso tal como lo hicieron; esta tesis junto con todo lo que se tiene de tecnología al alcance hoy, simplemente no existiría y tal vez estaríamos moliendo trigo con piedras y cazando con palos y flechas, tan solo para subsistir, o no.

5.2 Tipos de Canales

Se podría clasificar la transmisión del lenguaje en dos tipos muy diferentes de canales: el vocal y el escrito o textual. No entraremos aquí en lenguajes de señas ni otras cosas raras y de menor frecuencia de uso, sin desmerecerlas en lo absoluto.

Canal Vocal

El ***canal vocal*** se basa en que la persona emisora *piensa* un texto, luego lo dice (*lo emite mediante su modulador: el aparato fonoarticulatorio*) y el 'mensaje' convertido en una secuencia codificada de sonidos en el tiempo, llega a la otra persona en forma auditiva (*no consideramos en esta etapa los demás canales paralelos como contexto, gestos, prosodia y entonación, etc.*).

Finalmente la persona destinataria del mensaje, "entiende lo dicho" que no es más que una transmisión de "palabras" mediante un medio físico: usando codificación auditiva, por medio de un canal de sonido *(el aire u otro canal electrónico que permita transmitir la voz)* llegando finalmente a la mente de la otra persona, mediante un decodificador natural compuesto por el oído y su posterior reconocimiento de voz del destinatario.

Analizando las partes, simplificadamente y en ausencia de ruido, sería como lo siguiente:

Mente de Sujeto Emisor (IDEAS)
=> Lenguaje Natural (palabra1 palabra2 palabra3..)
=> modulador vocal (*aparato fonoarticulatorio*)
=> ***canal vocal*** (aire, teléfono, radio, TV, grabación diferida, etc.)
=> demodulador (*oído + reconocimiento del habla*)
=> Lenguaje Natural (palabra1 palabra2 palabra3..)
=> ***Mente del Sujeto Receptor (IDEAS)***

Analizando brevemente el mecanismo, se identifican 2 elementos importantes:

El ***emisor*** y el ***receptor***, que son la mente de ambas personas involucradas, conectados mediante un mecanismo de modulación y demodulación acústico, usando ***símbolos fonológicos*** que son secuencias de segmentos acústicos, llamadas palabras.

Canal Textual

El ***canal textual o escrito*** se basa en que la persona tiene ***una idea***, *piensa* un texto que la represente en un lenguaje natural que conoce. El proceso de transformación de una idea en palabras se conoce como lexicalización de una idea mediante una gramática de un idioma, es un proceso complejo de índole netamente cognitiva, el cual se ha estudiado y descripto en numerosos tratados, mas no se sabe como funciona exactamente.

La nueva ciencia que lo estudia, para recrear la generación de texto artificialmente se llama *Generación de Lenguaje Natural* y posee un grado de complejidad y profundidad

importante, incluyendo mecanismos de solución a problemas NP-duros[12] con algoritmos similares a los de aprendizaje automático en inteligencia artificial como ser Planning (*GraphPlan*) entre otros varios.

Luego de lexicalizado el texto, la persona se la 'dicta' internamente, deletreándola conforme a su nivel cultural y sapiencia, accionando algún sistema de codificación de símbolos que use caracteres y letras de un alfabeto acorde al idioma.

El mecanismo de selección secuencial de símbolos puede ser desde manuscrito mediante el uso un dispositivo de escritura físico/digital, mecanografiado y hasta táctil o gestual ante una cámara que reconoce gestos tipo *Amslam*[13].

Luego, con el estado actual del arte, el mensaje suele entrar en algún sistema de transmisión o memorización electrónico o digital (*Télex, Módem o Sistema Digital, etc.*)

En algún momento posterior, este mensaje en forma de secuencia de caracteres codificados de algún modo, llega a otra persona (*el receptor*) representados por algún dispositivo en forma visual (*pantalla o impresora*) y finalmente la persona mediante un mecanismo de percepción óptico llamado *visual* el cual posee una gran capacidad de decodificación natural por reconocimiento simbólico, llamada *lectura.*

La persona finalmente reconoce y clasifica con la vista los símbolos y su secuencia, 'ensamblando' luego palabras en su mente. Finalmente, accediendo a su léxico interno y mediante el uso de su conocimiento de las mecánicas y reglas de gramática en el idioma aprendido decodifica el mensaje y finalmente: *"entiende lo dicho" (IDEA)*

5.2 Modelo Humano de Transmisión de Texto

El conjunto de etapas o procesos, lo podríamos resumir en algo llamado "*modelo de trasmisión textual humano*" el cual procederemos a desmenuzar para identificar los mecanismos e intentar modelizarlos apropiadamente.

Antecedentes

Según *S. Deorowicz* [82] la modelización, detección y corrección de los errores en idiomas altamente flexivos, como su polaco nativo al igual que el español, entre otros; poseen serios problemas de recursos computacionales.

Analiza los modelos mediante una cadena de procesos físicos/mentales con resultados intermedios del siguiente tipo, Fig2.

Fig 2 *Modelo de Generación de una palabra escrita*

[12] Ver problema ***NP-duro*** en el glosario al final del trabajo.
[13] Nombre del lenguaje de señas para hipoacúsicos, llamado: "*American Sign Language*"

Modelo Propuesto

Una comunicación que involucra texto, no es más que una transmisión de ***una idea X*** ubicada ***en la mente*** de una persona **A** (*el emisor*), hacia una persona B.

Inicialmente usa una codificación de ***lenguaje natural*** a ***palabras***, (*usando verbalización mediante un léxico*) luego de palabras a simbólica (*símbolos y letras*), usando un canal de deletreo pictográfico vehiculizado de algún modo *(papel, video, o tipeo mecánico u otro electrónico que permita transmitir texto)* llegando finalmente a la mente de la otra persona, mediante un decodificador natural llamado *lectura*, compuesto por su vista y el posterior reconocimiento cognitivo de los símbolos junto al ensamblando en ***palabras*** del mensaje en ***lenguaje natural***.

Finalmente, la persona **B** (*el receptor*) puede recibir el mensaje para entender ***en su mente*** el resultado de la transmisión de aquella ***idea X***.

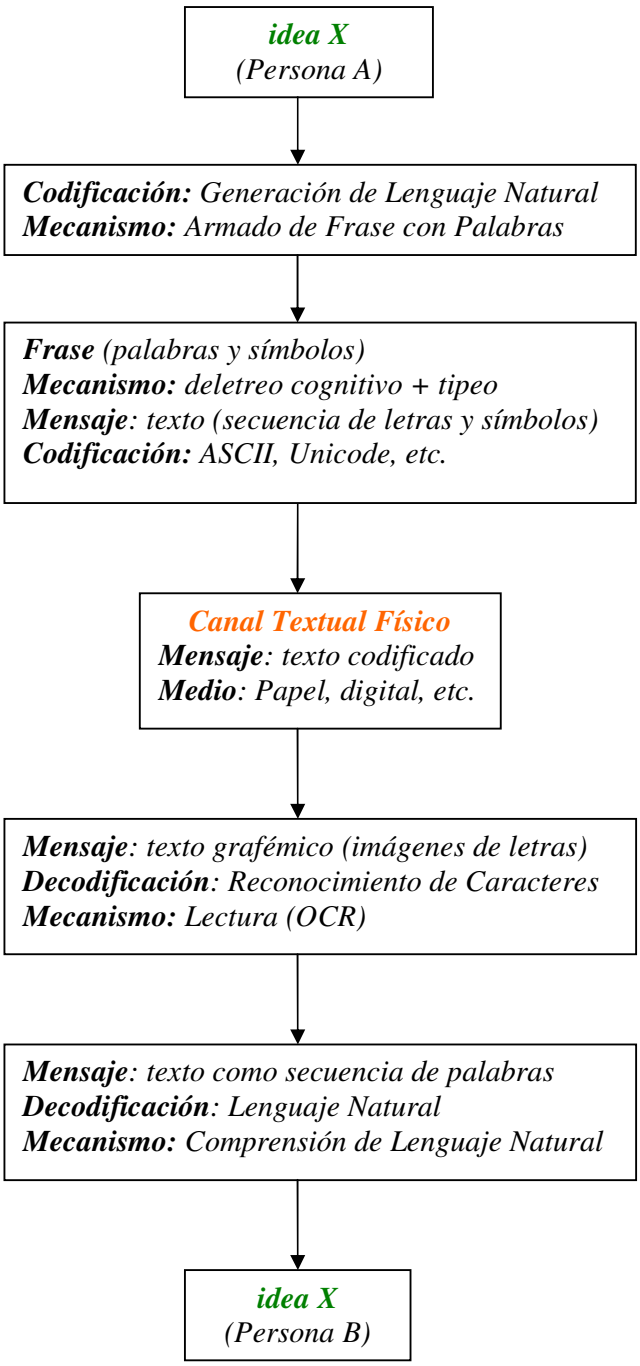
Hemos realizado un diagrama aproximado en la ***Fig. 3***, del proceso descripto anteriormente como un sistema de transmisión en etapas de elementos cognitivos que llamamos ***Ideas*** entre dos personas **A** y **B**, usando un mecanismo que pasa por texto escrito.

Se ha tratado de destacar y desmenuzar en conceptos de: mensajes, medios, tipos de codificación y decodificación, etc.

Diagrama

Este diagrama no pretende ser exacto ni exhaustivo, solo es un modelo para aislar determinadas secciones que nos interesan y analizaremos, pues si se desmenuzase aún más, se podría llegar hasta el análisis de los canales de reconocimiento lumínico en la retina, cosa que no tiene sentido en este trabajo y pertenece a áreas como fisiología de la visión y neurociencias, etc.

Fig. 3



Modelo desmenuzado de comunicación mediante palabras escritas entre personas

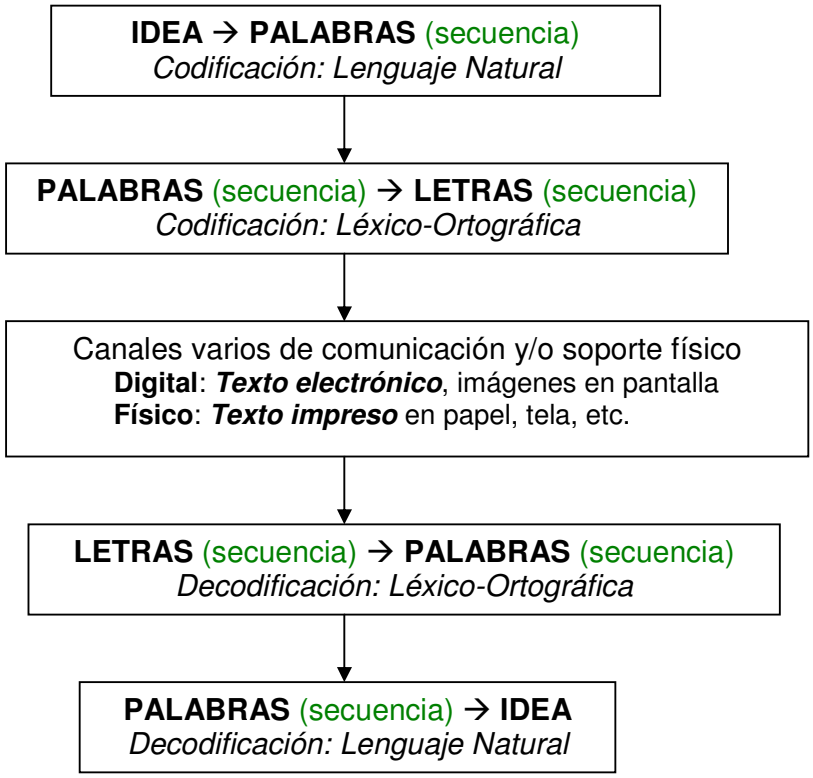
La única diferencia que puede observarse entre el ***canal textual*** y el ***canal vocal***, es de modalidad y no de fondo: los mecanismos de este tipo, en el caso de texto, no suelen ser en tiempo real.

Una de las pocas excepciones son los sistemas de transcripción en vivo o de ***chat*** en donde se vean las palabras en forma instantánea en el extremo receptor, mientras se ingresan en el extremo emisor, pero éste no es el tema que nos interesa aquí.

Elementos Constitutivos

Las partes comunes del sistema, ante la visión de la ingeniería de comunicaciones, son claramente un conjunto variado de codificadores y decodificadores en cascada. Este esquema simplificado se muestra en un nuevo diagrama en la ***Fig 4***

Fig. 4



Elementos constitutivos de comunicación de IDEAS entre personas

Si bien lo anterior parece demasiado obvio para cualquiera, ya que es un tema cotidiano y lo ejercitamos aún mientras leemos este mismo informe, se considera útil para poder identificar las partes a modelizar.

Análisis del Modelo

La teoría de transmisión y recepción de comunicaciones, en especial en el ámbito de la ingeniería electrónica, estudia y diseña desde los métodos de codificación y decodificación hasta los diversos modelos de los medios físicos de transmisión y

almacenamiento de la información y junto a toda la mecánica asociada, con el fin de optimizar, detectar puntos problemáticos, hallar, medir y corregir los errores.

Fundamentalmente, se busca tener bajo control un tema complicado y molesto, los errores de comunicación por diversos motivos, conforme a los modelos involucrados.

Para esto se analiza su generación y se pone mucho esfuerzo en diseñar mecanismos de modulación y demodulación robustos a fin de minimizar la aparición de errores.

Conjuntamente se diseñan estrategias y mecanismos de detección y corrección de aquellos errores que escaparon a la optimización y al diseño mismo.

En el caso presentado durante esta sección, que es la redacción, escritura y lectura, la mayoría de los mecanismos son naturales y nos involucran directamente, pues somos tanto decodificadores como codificadores y ejecutores en esta larga cadena de análisis presentada, por lo cual hay que tener sumo cuidado para ser imparciales y no estudiar esto en forma sesgada, dando por obvio cosas que hacemos sin darnos cuenta o automáticamente.

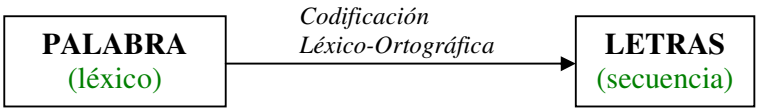
5.3 Errores de Comunicación

Quien desarrolló estos enfoques, basado en teoría de información por los años 40 fue *Claude Shannon* [31], pionero de las teorías de transmisión de datos e información, quien echó luz sobre la información misma, a partir de teorías claras y simples.

Para repasar todo análisis de una comunicación, comenzaremos por plantar el modelo de funcionamiento y desmenuzar las etapas donde ocurren cosas, para luego analizar los tipos de errores que puedan ocurrir en cada una.

Luego determinaremos cuales de esos errores podemos modelizar efectivamente, para finalmente concentrarnos en proponer modelos para lograr detectarlos e intentar corregirlos del mejor modo posible.

Vamos a enfocarnos inicialmente en un solo sector de todo el proceso anteriormente descripto, donde hay una propensión conocida y alta de errores de transcripción de texto:



Este segmento de codificación es naturalmente complejo y consiste de numerosos métodos diferentes de codificación mecánica *(teclados, mecanismos de tipeo de caracteres*) y un método de codificación cognitiva, del cual solo podemos modelizar y analizar su descripción en base a los resultados, ya que no hay como intervenir en él, salvo el obtener una idea mediante imágenes de ***MRI*** *(Magnetic Resonance Imaging).*

Las imágenes de ***MRI*** dan a los neuropsicofisiólogos información muy precisa de cuál área del cerebro procesa (*se activa o ilumina*) ante qué clase de estímulo y en que momento. La determinación de su localización precisa en el cerebro en forma no invasiva y a lo largo del tiempo, permite crear y ensayar diversas teorías de cognición, como la formación del lenguaje en el área de Broca, del acceso al léxico. Pero aún no se tiene el modelo definitivo y claro de su funcionamiento.

A ciencia cierta, es como tratar de saber adonde piensa llegar un taxista de Nueva York, yendo una noche por la 5ª avenida entre las calles 34th y la 33rd, tan solo viendo las luces de su auto desde un satélite, junto con todo el tránsito y sin poder distinguir cada luz, tan solo saber el total de brillo en el tiempo. Es un poco hasta naif el imaginar que, al menos con la tecnología actual eso se puede hacer así, habrá que esperar nuevos desarrollos.

Palabras en la Mente

Desmenuzando un poco más el bloque que nos interesa y será estudiado, veremos que las palabras están ubicadas en uno o más lugares en la mente de la persona y si bien esta definición es un poco difusa, no necesitamos más detalles.

Para no entrar de lleno en otra área, vamos a modelizar el fenómeno mediante una descripción simple: las palabras "suenan" por lo general *en la mente de las personas*.

Hay sobrada evidencia de ello [86], debido a que hay una dualidad en su génesis: las palabras y su genealogía son netamente de carácter fonológico-auditivo, por lo cual el "sonido" es lo que más representa la palabra en la mente e interviene en el aprendizaje.

Todo lo antedicho, sin duda permite inferir un sinnúmero de mecanismos útiles; tanto para detectar los errores de ortografía, como para poder predecir su reparación. Esto lo veremos paso a paso, a medida que se avance en el presente trabajo.

Sin duda hay un enorme número de otras teorías que intentan describir el acceso al léxico interno, la génesis de las palabras mismas, su interpretación, la capacidad de reconocer palabras nuevas por similitud semántica, morfológica y hasta fonética, etc. mas no entraremos a analizar estos aspectos aquí.

5.4 Errores en Palabras

Cuando una palabra posee errores, es análogo a la recepción de un mensaje con errores; el cual fue transmitido correctamente y hubo una transformación en el camino que cambió algo, que podemos llamar ruido simbólico parcial, pues afecta solo parte de las palabras. Precisamente esto es lo que pretende ser detectado y corregido.

Detección de Errores

La única manera de saber cuál es el error en una palabra arribada de un sistema de comunicaciones, es compararla con la palabra o mensaje que se transmitió originalmente y debiese arribar en su lugar.

Si bien detectar el error, parece una tarea trivial, no lo es en absoluto cuando solamente se tiene la palabra 'errada', que llamaremos ***palabra con error***; el error en una palabra se asume por su inexistencia en un diccionario, o por una falla a un nivel más alto de comprensión: gramatical o semántico, el cual no abordaremos en el presente trabajo.

En realidad la ***palabra original*** estaba *en la mente de la persona que creó el texto*, en un momento incierto y es sin duda anterior a su recepción, por lo cual esta información es imposible de obtener, pues ya no existe ni la palabra, ni se puede repetir el contexto.

Tampoco existiría forma de recuperar la ***palabra original***, en este otro caso; pues si tuviésemos a la persona disponible en un dispositivo de imágenes MRI, no podríamos con tecnología de hoy, hallar en que parte de su mente y cuándo se gestó cuál palabra.

Método Plausible

Volviendo a las *cosas que sí se pueden hacer*, como no podemos por ningún método existente saber *qué pensó la persona*, solo podemos especular con el acto consumado de la palabra que llegó escrita: tratar de deducir si está bien escrita o no y en todo caso, tratar de corregirla lo mejor posible, ***adivinando*** qué había en la mente de esa persona con la *evidencia forense* que tenemos: *la palabra errada* y eventualmente su entorno.

Para esto debemos analizar con cierto cuidado los modelos un poco inciertos, basados en lo que pasa en la mente de las personas, cuando se exterioriza un texto que termina escrito; para luego saber lidiar la clase de errores que nos podemos hallar en las palabras y textos para clasificarlos, lo cual haremos en la siguiente sección, en donde se analizarán hipótesis de los diversos errores y se atribuirán a las partes del modelo.

Todo esto influye sin duda en cómo definir la métrica y qué cosa contabilizar para medir la bondad y comparar exitosamente los diversos sistemas existentes del estado del arte con el nuevo desarrollo propuesto en esta tesis.

Para esto debemos saber con qué clase de cosas lidiamos, por lo cual clasificar los errores se torna absolutamente necesario.

5.5 Clases de Errores

En esta sección clasificaremos los diferentes tipos de errores que consideramos cuando se escribe un texto y no se tiene éxito con la ortografía.

Enunciaremos y analizaremos brevemente los diferentes tipos de errores de escritura y especularemos cuándo es probable que éstos ocurran, con el fin de tratar de determinar el tipo de estadística y dentro de ella, la prevalencia que éstos podrían tener.

Según numerosos autores [60] [65] [82], se pueden distinguir inicialmente dos clases de errores en palabras: los de *ortografía* y los de *tipografía*.

Numerosos autores coinciden en que es deseable que un procesador de texto pueda indicar y/o corregir la mayoría de estos errores, junto a otros errores de índole más cognitiva y abstracta, relacionado con el lenguaje y la gramática como lo son los de *concordancia, estilo y redacción*.

Hoy, los procesadores de texto, apenas se acercan a proponer palabras alternativas y los más avanzados o conocidos como el MS-Word, aún juega a las zancadillas con sus intentos de corrección, dado que poca gente activa la corrección automática en ellos.

Si bien hay módulos complejísimos para la resolución de errores de casi todo tipo, aún les falta bastante camino por andar para ser comparables con un corrector humano.

Recientemente han salido algunas ofertas de correctores de ortografía, incluyendo gramáticas y estilo; ellos están basados en web con tecnología on-line[14], en lugar de usar programas con paquetes auto-instalados como el Office o el Word. Éstos no fueron evaluados ya que no permiten interacción con una API y no poseen especificaciones.

Clasificación de Errores

La clasificación de los errores, que no es la única, los agrupa por el tipo de efecto y origen (es multifactorial) y serían del tipo Ortográficos y Tipográficos, dejado algunos otros en Tipo Superior, que se podrían incluir entre los Ortográficos, pero los vamos a tratar por separado.

Errores Ortográficos

Son errores complejos, de origen mayormente cognitivo, que suelen darse cuando el autor no sabe correctamente deletrear una palabra que escribe, o se le olvidó cómo se escribe, o simplemente se le borro su ortografía al escribirla.

Es importante analizar que cuando aparece este tipo de error, la palabra escrita se parece mucho a la palabra original, en especial fonológicamente, es decir al leerla y pronunciarla, resulta en algo similar a la palabra correcta. Esto último los torna difícil de detectar al releer un texto, aún para un humano.

[14] www.stylus.com

Una consecuencia de ello es que estos errores son fuertemente dependientes de las relaciones entre un determinado lenguaje a nivel letras y su transcripción a fonemas.

Errores Tipográficos

Son todos los errores asociados al método de tipear o deletrear mecánicamente las palabras, y son dependientes del tipo de dispositivo o teclado en donde se ejerce la transcripción. Ellos no afectan únicamente a las letras sino también a los signos de puntuación, motivo de numerosos errores en la detección de los límites de las palabras. Suelen ser originados por imprecisiones psicomotrices, distracciones, fallas en los mecanismos detectores de los accionamientos, entre otras múltiples posibilidades, algunas al azar. Otros errores son asociados tanto a los dispositivos como a las técnicas de ingreso de texto, como ser, en los táctiles tipo capacitivos y resistivos, la presencia o no de feedback mecánico tipo vibración en tabletas y celulares, reconfiguración del teclado automáticamente conforme a contexto *(Android KitKat, IPhone 2+, Blackberry Z10)*, como así la predicción interna estilo *T9*, entre otros numerosos motivos.

Errores de Orden Superior

Son los de *concordancia, estilo, redacción, retórica, etc.* los cuales no se abordarán por ser posteriores y de otro nivel, en la cadena de procesos e involucrar disciplinas hoy bastante ajenas a la ingeniería como la lingüística, la gramática y la filosofía, a pesar de que las primeras dos, son cada vez tratadas por más matemáticos e ingenieros.

Antecedentes sobre Técnicas para el Tratamiento de Errores de Texto

Numerosos autores [80] [74] [56] estudiaron los errores de texto en profundidad y propusieron técnicas mixtas y muy variadas, para su corrección. La mayoría de estas técnicas están adaptadas para cada idioma y por ende son extremadamente particulares.

Distribución y Estadística de los tipos de Errores

Los autores *B. Berkel y K. Smedt,* [41] afirman que el 80% de los errores del tipo tipográfico se pueden presumir como ocasionados por simples operaciones de edición como inserciones, falta, sustituciones y transposiciones de letras simples; mientras que el restante 20 % de errores son por causas más complejas, lo que fue observado en 1980 por *Peterson J.L.* [46]. En esa década hubo numerosos intentos [64] en detectar y corregir automáticamente errores en los textos.

5.6 Etiología de los Errores de Texto

Enunciaremos con una breve descripción cada tipo de error y luego para estudiar su origen o identificar su etiología, enunciaremos una breve hipótesis de su probable génesis con el fin de tratar luego, de diseñar una heurística que nos permita detectarlo.

Además, el tener un conocimiento cabal de los orígenes y la estadística asociada a su aparición puede sin dudas, ayudar a concebir un método para su corrección.

Evidencia Estadística de los Tipos de Errores

Los errores de texto, como ya se enunció antes; son un 'hecho consumado' y lo único que se tiene para determinarlos es el texto resultante sobre el cual solamente se puede realizar un análisis forense.

Los métodos forenses para su detección y análisis, varían desde comprobaciones directas automatizadas con listas de palabras, hasta modelos estadísticos avanzados combinados con heurística que logran identificar, además de los errores ortográficos puros, los errores de estilo, concordancia, gramática y varios otros; arrojando cifras cuantitativas cuya interpretación es tan falible como los errores mismos.

Sin duda la comprobación última y de mayor calidad es la humana, realizada por expertos; pero ésta lamentablemente es muy costosa, de lenta ejecución y no puede manejar los volúmenes de texto necesarios para lograr una significancia estadística útil.

Se puede afirmar, dado que los errores de ortografía son de tantos tipos y orígenes, que los análisis contables hechos de los mismos sirven en la medida que contemplen modelos claros y repetibles de su gestación, que condigan con ellos y se verifiquen.

Esto mismo motivó a un número importante de investigadores de todas las latitudes, a evaluar el tema mediante herramientas de variadísimo tipo; esto lo veremos brevemente a continuación, en un extracto.

Antecedentes y Estadísticas

Cuando hay un fenómeno, la ciencia primero cuantifica, luego clasifica y finalmente analiza los posibles orígenes creando teorías que debiesen de condecir con los datos y fenómenos observables o medibles.

Numerosos trabajos mencionados en [65] sobre la evidencia de los errores de tipeo en escritura, concluyen que el 80% de los errores son de una sola operación de edición, el 20% restante son errores de origen cognitivo-fonético de título más complejo.

Análogamente *Pollock y Zamora* [73], en 1984 hacen un análisis sobre errores en textos escritos y analizando 50 mil errores de un corpus con 25 millones de palabras en idioma inglés. Ellos concluyen que alrededor del 72% son de 3 tipos simples: reemplazo, falta o inserción errónea de una sola letra, aunque no analizan el origen presunto del error.

Los investigadores y lingüistas *F. Bustamante y E. L. Díaz* [75] en una publicación científica de *Microsoft Research*, reportan cifras similares en la distribución de errores y su etiología, trabajando sobre corpus propios de 8 millones de palabras en español. Ellos hallan que la mayoría de los errores se deben a uno de 6 tipos: omisiones/cambios, típicamente falta de acentos, substituciones y falta de mayúsculas, errores cognitivos, adiciones, substituciones directas y transposiciones. Concluyen que el porcentaje de los errores en la primera letra (~5%) y que los errores por adyacencia del teclado son de

menor cuantía, son mayores en el español que los reportados para el inglés; destacando que la mayoría de las estadísticas de errores que se publican son para el inglés y que no son extrapolables ni aplicables a otros idiomas más flexivos como el español.

La Dislexia

Recientemente en 2012, *Luz Rellos y B.Yatess* [74] reportan una alarmante cifra de población con dislexia de entre el 8 y el 11% en los estados unidos, basados en estudios de textos de la web 2.0 tanto del español como del inglés. Ellos clasifican los errores en dislexia fonológica y dislexia grafológica, con similar incidencia.

Estos estudios confirman que parte de los errores de ortografía son endémicos de la clase humana y responden a problemas cognitivos, de acceso y expresión del léxico; más que a problemas puramente mecánicos y/o estadísticos.

Hay trabajos *Pedlar* [83] en donde se abordaron la detección y corrección automática de estos tipos de errores en inglés. La investigadora *Luz Rellos*[15] hizo un estudio de estos tipos de errores, recopilando datos y acuñando el término "*dyswebxia*" para problemas relacionados con dislexia y la web.

Este tipo de errores de carácter cognitivo son excepcionalmente complejos de detectar y en consecuencia corregir, ya que suelen poseer patrones de mímica fonológica, los cuales son difícilmente atacables por algoritmos estadísticos puros, por tener reglas poco claras o complejas.

Los Idiomas y sus Errores

La mayoría de los trabajos citados estudian los errores en idioma inglés, algunos pocos en alemán, francés y escasos otros idiomas. Algunos de los trabajos citan que el español y los idiomas más flexivos poseen una mayor dificultad de restauración ortográfica y de hecho la restauración diacrítica se trata numerosas veces como todo un tema separado, siendo por ejemplo, inexistentes las marcas diacríticas en el idioma inglés.

La restitución de mayúsculas/minúsculas en especial cerca de puntuaciones, en siglas, en abreviaturas y con nombres propios, es un problema universalmente muy difundido por su dificultad. Esto no afecta mayormente la comprensión del texto sino el estilo y la elegancia, salvo contados tipos de ambigüedades en donde la colocación de una mayúscula pueda diferenciar entre un nombre propio y una palabra gramatical.

Errar es Humano

Queda poco claro la real gestación del error y su etiología, ante tantas teorías, aunque lo único real y tangible es su presencia inexorable en prácticamente todos los textos; si bien más y más personas utilizan hoy los sistemas informáticos, la existencia de los errores y frecuencia no disminuyen, muy a pesar de los numerosos sistemas de asistencia para su corrección, habiendo evidencias de que una parte importante de su

[15] http://www.luzrello.com/Dyswebxia.html

origen, viene embebida dentro del intelecto humano, clasificado como dislexia [74]; concluyendo, como proclama un dicho latino muy antiguo: "*errarum humanum est*"

Enumeración de los Errores y sus tipos

Vamos a mencionar no solo errores cognitivos (*ortografía*) y de transcripción mecánica (tipográficos) sino algunos otros fenómenos combinados, observados en los textos escritos. Éstos, a nuestro entender, deben ser tomados en consideración, dado que se pretende emular la capacidad humana de lectura de texto con errores; en donde se presentan fenómenos asociados a la lectura y a la cognición como un modelo de reconocimiento enriquecido.

Errores involuntarios por distracción **(tipográficos)**

- Inserción de una letra adicional que no corresponde al lugar, presumiblemente es por un acto fallido o el accionamiento de dos teclas contiguas por error.

- Omisión o falta del tipeo de una letra, se presume que tal vez la persona supuso haber percibido el feedback táctil llevándolo a pensar que se ingresó, o tal vez el feedback del teclado es imperceptible o es del tipo táctil-resistivo, y la presión aplicada no fue suficiente y no provee feedback mecánico, entre otros tantos motivos posibles.

- Tipeo (ingreso) erróneo de una letra, en lugar de otra, sin motivo aparente alguno, suele estar asociado con desordenes cognitivos como la dislexia o simplemente por distracción, provocando este tipo de 'acto fallido'.

Errores de distracción de origen mecánico **(tipográficos)**

- Tipeo de una letra "cercana" en un dispositivo de ingreso mecánico (teclado) por distracción y/o falta de precisión mecánica del acto voluntario de tipeo.

- Ingreso "múltiple" de una letra por falta de percepción del acto consumado mediante un sonido, 'feedback' mecánico: vibración, clic, etc. Esto es mucho más frecuente en los teclados táctiles capacitivos que en los resistivos. También se da cuando el tiempo de depresión (*mientras está accionado*) supera, por distracción, cierto valor y el controlador del dispositivo comienza a 'repetir' reiteradas veces el carácter, anticipando una posible intención de repetirlo.

Errores asociados al método **(tipográficos)**

- Ingreso del carácter cercano "de al lado" por resultar superficies o teclas muy pequeñas vs. el tamaño del dedo (celulares y teclados pequeños). Es también asimilable a una distracción, mencionada anteriormente.

- Ingreso de otro carácter del grupo de letras asociado a un número en un teclado numérico de un celular, actuando por multi-tap *(clásico ingreso con teclados reducidos*)

- Aceptación errada de una palabra "sugerida" usando mecanismos de predicción automáticos o con aprendizaje, en este caso las palabras estarán ortográficamente bien escritas, salvo que se le haya enseñado mal la palabra.

- Errado de las posiciones y velocidad de deslizamiento en caso del método *Swipe (método novedoso de ingreso en teclados táctiles que consta en deslizar un dedo sobre las posiciones de las letras sin levantar el mismo)*

Errores mayormente cognitivos (ortográficos)

- Ingreso de caracteres similares en forma, por ejemplo la letra '**o**' y el cero '**0**' que se ven iguales (*de reojo*) y están demasiado cerca en los teclados 'qwerty', constituye un error muy frecuente y difícil de detectar visualmente durante la vorágine de una escritura veloz, siendo esta detección dependiente a la vez del tipo de tipografía o 'font' utilizado. Por esto en computación se utiliza mucho el cero tachado con caracteres mono-espaciados, para diferenciar claramente el número de la letra 'o'.

- Confusión de letras por sonido fonético similar, como los son las letras 'n' y la 'm', las cuales para empeorar las cosas están contiguas en los teclados '*querty*'. Lo mismo ocurre con las letras 'b' y la 'v'. El intercambio de las letras 'z' con la 's' es menos frecuente, pero es bastante común la confusión 'y' con la 'i' la cual también es atribuible a este mismo fenómeno, a pesar de no estar muy contiguas en el teclado '*qwerty*' (*están en la misma fila, pero salteando una letra*). Otro par de letras complicadas son las fricativas, como ser la 'c' y la 's', a pesar que la letra 'c' puede resultar plosiva, es decir sonar como la letra 'k', frente a determinado tipo de letras.

- Inversión de 'pares de letras' en especial cuando están en lados opuestos del teclado y el humano tipea con ambas manos. Se presume que el cerebro manda la orden, partiendo del léxico interno (área de *Brocca*), en la secuencia correcta. Como la velocidad de respuesta del sistema cerebelo-muscular del lado derecho e izquierdo puede diferir *(no somos simétricos y los diestros son más veloces con el lado derecho, se presume que por el alto entrenamiento de sus redes neuronales, deviene en mayor destreza y definición del diestro o zurdo, según el caso)*; da como resultado la inversión de pares de letras. Por lo general esto, según lo observado, ocurre al final de las palabras cuyas terminaciones son muy frecuentes en muchas palabras del corpus, o en partículas cortas de alta frecuencia en corpus. También pasa más seguido con letras cuya frecuencia es muy alta en muchas palabras, o esas palabras son muy frecuentes. Por eso el cerebro, que ya las conoce y está muy entrenado, baja los tiempos de respuesta o disparo y las manda demasiado 'cerca' unas de otras en el tiempo y finalmente se invierten mecánicamente por esas presumibles pequeñas diferencias en los tiempos de respuesta del lado derecho e izquierdo. En particular a mi persona y a bastante otra gente, nos pasa con *'que'*, lo cual solemos escribir *'qeu'* del mismo modo que *'los'* se escribe erradamente: *'lso'*. Otros pares conflictivos son *'la'*, *'le'* y *'ro'*
  Aporta un poco más de evidencia que los pares de letras (*bigramas*) de muy alta frecuencia y que se escriben con la misma mano por su posición en un teclado

'qwerty', no aparecen como errores frecuentes, como por ejemplo los pares de letras: *'ui', 'po', 're', 'as', 're', 'er', 'de', 'ju', 'ji', 'oi'.*

- Falta de conocimiento de reglas de ortografía: confusión de *'mp'* con *'np'*, omisión de acentos ortográficos o escritos, falta de colocar tilde (crema) sobre la 'n' para formar la 'ñ', confusión del tipo de acento en idiomas como el francés, polaco, noruego, etc. en donde hay diversos acentos y marcas diacríticas. Por ejemplo los acentos agudos y graves que se ven muy parecidos, solo que apuntan hacia atrás o adelante, cosa difícil de divisar en una pantalla si el texto es pequeño o si el lector no tiene suficiente agudeza visual.

- Originados por dislexia [74], una afección del tipo cognitivo de origen neurológico que afecta al 70-80% de la fracción que presenta discapacidades en el aprendizaje del lenguaje que es un 15-20% de la población mundial. El tipo de errores producidos por esta afección esta clasificado en dos grandes grupos diferenciados: dislexia-fonológica y disgrafia (*errores al escribir grafos/texto*). Si bien hay controversia en el aislamiento de las fuentes probables de los errores, con origen en la dislexia; esta última está claramente definida como ***"desorden de lectura y desorden de expresión escrita"*** en el DSM-IV [16]; siendo a la vez definida como ***"desorden de lectura y deletreo"*** en el ICD-10[17]
  En otras palabras los problemas en la lectura y comprensión de texto pueden ocasionar dificultades en su escritura y no se pueden aislar pues la lectura participa activamente en la escritura a nivel cognitivo. El resultado es un conjunto difuso de errores del tipo cognitivo presentes en una importante fracción de la población, generando errores tanto fonéticos como grafémicos, de difícil reconocimiento.

Errores voluntarios (producidos adrede)

- ***Reemplazo Fonético Aproximado***
  Otro error común, que más bien es un recurso extraordinario atribuible a la creatividad cognitiva, es el reemplazo fonético aproximado: como el poner 'ni' cuando no se tiene el recurso de la 'ñ'. Escribiendo 'anio' en lugar de 'año'. Otro mecanismo es utilizar el símbolo numeral '#' para reemplazar letras inexistentes en el mecanismo de ingreso, como se hace en algunos sistemas para escribir *año*, por falta de la 'ñ' ponen *a#o*.

- ***Números por Letras***
  Reemplazo accidental o voluntario de letras por números de forma similar, por ejemplo el número '3' suele ser usado para reemplazar una letra 'E' a pesar de estar espejada horizontalmente, el número 5 se puede interpretar como la letra 'S' al igual que el 2 puede ser visto como 'Z', el '8' como 'B' y mucho más frecuentemente de lo esperado el cero '0' es confundido con la letra 'O' y el uno '1' con la letra 'l' (*ele*). El número '6' con la letra 'b', el siete '7' con la 'L' y la

[16] DSM-IV manual de *Diagnostic and Statistical Manual of Mental Disorders* (*Diagnóstico y Estadísticas de Enfermedades Mentales) (DSM-IV)* de la Asociación Americana de Psiquiatría (2000)
[17] ICD-10 *International Classification of Diseases* (*Estándar Internacional de clasificación de Enfermedades) (ICD-10)*

'h' con el '4' (*invertidas*) y otras sutilezas. De hecho en ciertas tipografías algunas de estas similitudes son indistinguibles a simple vista y hasta en algunas pocas, hay letras y números que son exactamente iguales. En computación se utiliza hace bastante tiempo el cero tachado, para evitar errores al codificar programas, ya desde la época en que se usaban tarjetas perforadas y tal vez antes. 5A1A (sala), *SALI0 (salio), d3cir (decir) cr3w (crew), 3773* (***ELLE*** *al reves*), 0TR0 (otro), S3AM0S (seamos), 5010 (solo), etc.

- Omisión de algunas o todas las vocales, para producir abreviaturas artificiales o para producir efectos fonéticos entretenidos, que distorsionen o guíen la lectura, por ejemplo saludos puede escribirse: *sld, tkm, pltd.*

- Otro clásico error de corte desde involuntaria hasta voluntaria, es la repetición de un mismo carácter, el cual a veces no se percibe fácilmente durante la lectura. Esto también es utilizado para enfatizar la palabra, o esa sección fonéticamente, en lugar de utilizar los diacritos (acentos o tildes escritos), por ejemplo muchas veces se pone: *chauuuuuu, hooooolaaaa! batatttta, perrrrrrrro*, etc.
  Si bien esto no constituye un clásico error de ortografía, genera un serio problema de reconocimiento por parte de una máquina y la idea central del presente trabajo es tratar de reconocer el texto de la misma manera que lo haría un humano.

Letras con Números **(producidos adrede)**

- ***Números Fonéticos*** Muchas veces las personas "juegan" con el lenguaje y logran ciertas combinaciones onomatopéyicas, que resultan en palabras conocidas, si se pronuncian independientemente y juntos los nombres de letras aisladas y los nombres de los caracteres o números o viceversa.
  Por ejemplo si se escribe lo siguiente, la interpretación es obvia puesto que al no existir esa palabra, se puede mentalmente expandir el número 2 fonéticamente, pronunciando "dos" y finalmente se componen ambas creando la palabra salu2 = salu+dos = saludos.

  salu2 → *saludos*

  Este mecanismo es sencillo de implementar y su aplicabilidad es interesante, dado que se resuelven numerosas onomatopeyas utilizadas en blogs, twitter y mensajes de texto. Esto también es usado en forma interesante cuando se tratan de resolver acrónimos inventados para abreviar palabras comunes, por lo cual también se debe tener en cuenta la posibilidad de expandir acrónimos y letras sueltas, en forma fonética tratando de encontrar palabras que se forman y tengan sentido. Ejemplos de este mecanismo son algunas palabras entre comunes, graciosas y hasta ofensivas, como ser: *salu2, p2, sali3, 2ar, pu3facto, si2o, senta2, ca3, ventila2, pin8, dientu2, agua0, si1000, para2, vac1, casa2, vien3.*

- ***CamelCasing Fonético*** usando letras en mayúscula dentro de la palabra para efectos fonéticos, en donde la letra en mayúscula se inserta con la intención de que su pronunciación sea textual, por ejemplo: *saliT (sali**te**) correT, Plado (**pe**lado), TneT (**te**ne**te**), Vcino (**ve**cino)*

Ruido y Basura **(adrede)**

- Escritura de no palabras en forma violenta o espástica generando 'ruido' como ser: *lskdflasihasdnlg, lsdvnwukgh*, (*como se diría de un gato saltando sobre el teclado*) Esto, a pesar de ser no aparente, aparece frecuentemente y se ha observado en numerosos blogs. Los motivos son desconocidos pero el costo o dificultad de este análisis es importante, ya que no posee una estadística asociada, más allá de la de todas las posibles combinaciones caóticas de letras.

- Inserción de pedazos de 'scripts' y cosas raras tomadas como texto, esto también se da con usuarios de chat, blogs, facebook. Se estima que lo hacen por error y/o a propósito y con intención incierta: copian y pegan trozos de cualquier cosa de la web o de sus computadoras: scripts crudos en HTML, XML, C++, Java, PHP, u otro Script, o insertan pedazos de datos puros vistos con un editor texto, 'snippets' (trozos) de lenguaje de máquina interpretado como texto, entre otros 'horrores ortográficos'. La detección de esta modalidad es compleja, un analizador morfológico hará todo lo posible en forma incremental, mas la falta de gramaticidad y la alta proporción de palabras irreconocibles y símbolos poco frecuentes en escritura, podrían ser usados para clasificar estos 'engendros'.

Escritura de texto 'elegante' *(con doble lectura)* **(adrede)**

- Uso de Mayúsculas y minúsculas (se llama Camel-Casing) con palabras pegadas, como el caso de los Hashtags de twitter, las direcciones de mail, algunos nombres de marcas, nombres propios compuestos, etc. Por ejemplo: *#HoyNoComemos AndresT@miMailServer.com RadioGaga*

- Inserción de guiones-bajos '_' para separar visiblemente palabras, ej: *guia_para_sordos, me_robaron,* etc.

Palabras Pegadas *(sin espacios ni puntuación)* **(tipográfico)**

- Muchas veces la falta de escritura del espacio o signo de puntuación que separe las palabras no nos permite diferenciar los límites de una palabra, para luego al cortarla en trozos y analizarlos individualmente.
  Ejemplos de esto serían ilimitados, pero muy frecuentemente se dan cosas así:

  10pesos ==> *10 pesos*
  sinduda ==> *sin duda*

  El motivo de esto puede ser distracción, causa mecánica o el hecho de haber sido cometido adrede para esconder palabras ofensivas o hacer criptografía tipográfica. Este tema será abordado porque hemos detectado su alta frecuencia de aparición en corpus.

Palabras que suenan iguales *(alófonos = ambigüedad fonética)*

- Muchas palabras, cuando son escritas y leídas o pronunciadas suenan exactamente igual, por ejemplo: **hola** y **ola**, **hasta** y **asta**. La lista es larga y

básicamente de lo que se trata es que muchas personas escriben una palabra tal como recuerdan su fonética y como existen ambas, aparecen bien escritas y el diccionario/corrector no acusará error alguno. Este es uno de los casos 'difíciles' y será tomado en cuenta para resolverlo, puesto que si un humano escucha una de esas palabras, tomará la decisión de cual acepción es cuando tenga algo más de contexto, y esto no es de una primera etapa del análisis y deberá ser comunicada a una etapa posterior, por lo cual sería deseable que el sistema de corrección entregue las opciones cuando existan o se sospeche la existencia de alófonos. Esta sería una de las virtudes de un corrector 'inteligente'.

Palabras que difieren en una letra (de más, cambiada o insertada)

- Este es un caso muy frecuente con palabras cortas, en donde por una letra puede cambiar completamente el significado. La probabilidad de que en palabras cortas con ese cambio la palabra resultante pertenezca al diccionario es mucho más alta que con palabras de 6 o más letras. Frente al ingreso de esta palabra si un corrector ortográfico clásico como el del MS-Word no le acusó error, es muy probable que no la vea ni siquiera al repasar. De hecho me pasa seguido y es uno de los problemas más frecuentes de desprolijidad.
  Este caso sin dudas es complejo de resolver. Resulta interesante presentar un número finito de opciones ortográficas para cuando la palabra en cuestión se encuentra en el diccionario. Para diferenciar entre las opciones presentadas incluyendo a la original, se crea una variable difusa asociada a cada una de ellas, que llamamos *Verosimilitud*, la cual indica mediante su valor, si la palabra presentada es la original o una opción diferente. Esta opción tal vez pueda ser la que el autor intentó escribir en el contexto de la redacción.
  Esta ambigüedad, podría resolverla una etapa posterior con mucha mayor inteligencia y teniendo en cuenta el contexto completo ayudada por esta variable.

Fenómeno de 'priming' con la primera y última letra (cognitivo)

- Cuando alguien lee de corrido, en contexto y entendiendo lo que lee, es muy frecuente que las palabras sean leídas correctamente si solo la primera y a veces la última letra están correctas, las demás pueden estar fuera de orden o ser parecidas grafológicamente.

- Otro fenómeno asociado al anterior es que la primera letra de las palabras, suele estar bien casi siempre, esto también es una observación útil a la hora de realizar correcciones especulativas. Se afirma que menos del 10% de los errores de ortografía por sustitución de una letra lo hacen efectivamente en la primer letra de una palabra, aunque diversos autores afirman que en idiomas como el español esta cifra es mayor.

Creando adrede confusión o dobles lecturas

- Hay cierto tipo de efectos ‘especiales’ construibles mediante tipografía, fonología y otros recursos unidos al ingenio humano. Sólo son detectables mediante una lectura inteligente y perspicaz, leyendo hasta ‘entre líneas’.
  Éstas son mayormente situaciones buscadas por el escribiente para comunicar

cosas en formatos '*elegantes*' y tal vez un poco extraños, inclusive son usados para burlar sistemas de control de contenidos ofensivos, hacer chanzas, hacer dibujos con caracteres, etc.

- Uno es el llamado en Argentina '*jeringoso*', que consiste en imitar a alguien con un defecto de tartamudeo raro, agregando a cada sílaba una nueva sílaba formada por la última vocal anteponiendo una letra plosiva 'p'. Por ejemplo codificando la palabra: ***murciélago*** obtendríamos ***mupurciepelapagopo***.
  Esto no es frecuente en escritura y se podría resolver de modo relativamente simple mediante patrones gramaticales de letras construidos con autómatas determinísticos finitos para su reconocimiento, pero no lo consideramos necesario pues es un fenómeno aislado, poco frecuente en textos y muy regional.

- Otro método es el utilizado para disfrazar textos mediante el uso de camelcasing/pascalcasing; por ejemplo, embebiendo mensajes ocultos mediante mayúsculas asi: *gran PUerTO DE MIsión ERa DAdo por cerrado.*
  Si bien esto debiese y pudiera ser fácilmente detectado, la idea es que generará un segundo mensaje en 'paralelo', y en ese caso no está claro como presentarlo. Esto mismo es como una especie de criptografía, que se puede lograr mediante cambios de tipografías, negritas, estilos, etc. La necesidad de la detección de estos efectos excede el concepto que se propuso en este trabajo y su utilidad solo pertenece a ciertos ámbitos con sutilezas, además de ser un fenómeno que concierne múltiples palabras, por lo que no se abordará en este trabajo.

Conclusión

Se han enunciado y analizado un conjunto amplio de errores presumibles y cambios en los textos, no compatibles con la gramática ni la ortografía.

Dado que estos errores están sujetos tanto a la creatividad como a la infinita estupidez, falibilidad y/o intencionalidad humanas, probablemente se haya omitido alguna pequeña parte de los posibles errores creables o existentes, siendo razón de esto que el autor también es humano y la perfección es solo divina.

Con los enunciados propuestos y analizados, afortunadamente se han podido aislar un conjunto interesante y útil de conceptos para poder postular y ensayar algoritmos de corrección eficientes y efectivos, descartando motivos que se asumen poco frecuentes.

Recordemos que la idea e intención principal que subyace es que el sistema creado debe hacer el mejor esfuerzo con lo que se sabe de ingeniería, lingüística básica y del tipo de error presumible, para poder extraer y presentar toda la información posible, inclusive la redundante, mientras se pueda realizar con una buena relación costo/beneficio, en términos de recursos computacionales y tiempo de proceso.

Perspectivas de Uso

En el caso de que nuestro sistema sea colocado en serie, previo al ingreso de datos de un sistema inteligente que procese luego texto en lenguaje natural; la idea es que este último reciba toda la información necesaria con valoraciones y métricas precisas, con el

fin de que pueda realizar la decisión más correcta, basada en un presumible nivel de inteligencia y/o en una lógica de más alto nivel de abstracción o meta-cognición.

Es importante que esas métricas y valoraciones estén definidas visiblemente y que expresen en forma clara, útil y práctica la información recabada.

Esto es lo que se abordará en detalle, en la próxima sección. Y posteriormente, en la **Sección 6.3** se vuelve a abordar el tema, esta vez citando autores con sus hallazgos para, finalmente, enfocarse en el tallado del algoritmo.

5.7 Métricas de Ortografía

Vamos a enunciar algunas definiciones para poder crear una o más métricas que nos permitan cuantificar algún valor asociado que sea útil.

Las palabras se codifican como ya hemos comentado, mediante códigos numéricos (*ASCII, Unicode, etc.)* y en computación se las llama ***strings*** (*cadenas*).

Existe una rama de la computación que trata justamente de operaciones con ***strings***, las cuales son muy variadas y hasta extremadamente complejas, llegando a aplicarse técnicas de aprendizaje automático para solucionar ciertas cosas como coincidencia de genes, pues toda la genética humana se codifica con secuencias de apenas 4/5 letras: "**`ATGCU`**". Lo complicado es hallar similitudes entre zonas de secuencias de millones de símbolos, esto es un problema abierto y suele ser *NP-duro*.

Mencionaremos solo algunas de las técnicas y métodos que hallemos más útiles para su aplicación en el problema de errores en lingüística y en especial aquellos aptos para el reconocimiento y tratado de los errores de texto, sin querer hacer un tratado de esto, que está en todos los libros específicos que abordan el tema. Ya hay trabajos de *Alberga* [57] que tempranamente se enfocan en la temática de la ***similitud*** entre ***strings*** y el error de deletreo (*spelling*).

Similitud y Distancia

Podemos aplicar en estos casos dos métricas diferentes a las palabras[18]: *similitud* y *distancia*, las vamos a definir en los próximos párrafos y mostrar porqué son '*casi*' inversamente proporcionales.

La distancia plantea una medida de cuánto se debe esforzar para convertir una cosa en la otra, donde 0 es ningún esfuerzo, lo cual las presume iguales, aunque esto no siempre es cierto. Por lo general es un número real, adimensional y no acotado.

En cambio la similitud, es un número acotado entre 0 y 1 que mide cuan parecidos son, para una determinada percepción, donde 1 es cuando son el máximo de similares y se podría afirmar que son 'perceptivamente' iguales, aunque no siempre son idénticas, mientras que similitud 0 sería equivalente a que son absolutamente diferentes, lo cual perceptivamente es un límite difícil de imaginar, pues implica una distancia infinita.

Por lo general si dos palabras tienen distancia 0, proseen similitud 100% y cuando tienen distancia muy grande, su similitud tiende a cero. Allí es donde la fórmula no es una inversa directa y depende del autor, puesto que no es posible siempre plantear con que coeficiente invertir las distancias para que resulten 100% y lleguen a 0 cuando son muy distantes y/o de diferente longitud.

---

[18] Esto no restringe que haya otras métricas, solamente se hace foco aquí en este par de métricas como distancia y similitud, aplicado específicamente a la lengua.

Distancia de Edición

Se llama '*distancia de edición*' a un número que representa el mínimo de *operaciones de edición* que han sido necesarias para transformar una *palabra A* (***string***) en otra *palabra B* (***string***).

5.7.1 Operaciones de Edición

Proponemos cuatro[19] tipos de operaciones simples de edición, las cuales enunciaremos:

- Cambio/Reemplazo de una letra por otra

  `pe**r**o → pe**c**o`

- Eliminación/Deleción de una letra

  `s**a**ldado → sldado`

- Inserción/Agregado de una letra nueva

  `invernadro → invernad**e**ro`

- Permutación de dos letras contiguas.

  `accionamie**tn**o → accionamie**nt**o`

Mediante estas operaciones, se puede transformar cualquier palabra en otra, eso está demostrado y es sencillo de inferir. El orden de estas operaciones para lograr una transformación no es único y a veces solo interesa el conteo del mínimo número de operaciones.

Distancia de Levenshtein

Es un algoritmo que halla el mínimo número de operaciones, para convertir cualquier palabra o *string* en otra. Se resolvió con un algoritmo que usa programación dinámica, el cual es eficiente, requiriendo un espacio de memoria y de operaciones de apenas A·B siendo A y B el número de letras de sendas palabras. A esta distancia de edición se la denomina distancia de edición de *Levenshtein* [42].

---

[19] Se puede demostrar que alcanza con apenas 3 tipos de operaciones: cambio, eliminación e inserción, para convertir cualquier palabra en otra de cualquier longitud, en consecuencia la permutación puede ser reemplazada por dos remociones seguidos de dos inserciones, que equivale a dos reemplazos, pero al ser una operación frecuente en errores de ortografía, puede resultar conveniente considerarla como una operación única.

Otras Distancias

Existen una gran cantidad de distancias de edición definidas entre *strings*, muchas de ellas son variantes sutiles de las otras, útiles en determinadas situaciones. Podemos enumerar las distancias de *Jaccard, Demerau-Levenshtein, Jaro-Winkler, Hamming,* entre tantas otras.

Todas ellas tienen diferentes enfoques y arrojan números con significado variado. Su utilidad no es solo cuantitativa sino a veces cualitativa y aplicadas a texto con errores solo sirven en la práctica para medir errores de pocas letras, en donde también el resultado confunde. Por ejemplo una distancia de 1, medida por el algoritmo de *Levenshtein* no nos dice mucho, pues el valor no especifica de cuál operación se trató, tampoco indica cuánto "*cambio*" la percepción entre las palabras por esa diferencia, en definitiva, no indica nada que parezca útil en el terreno lingüístico.

Lo importante y notorio es que estas distancias son monótonas y nos informan algo con apenas un número entero o racional, que puede ser proporcional al número total de cambios junto a la relación entre el número de letras de cada una. Pero no dicen nada del tipo de cambio que sucedió, siendo por lo general de escasa o nula utilidad como métrica asimilable con alguna percepción humana.

Hay algunas distancias que ponderan el tipo de cambio y cuáles letras hubo como par de reemplazo, usando el concepto de *Levenshtein* de distancia de edición, unido con alguna relación fonética. Si bien estas funcionan un poco mejor, tampoco logran una medida muy útil para este trabajo.

En la **sección 7** se presenta el algoritmo desarrollado y publicado dentro del marco de este trabajo, para enriquecer el modelo de restauración ortográfica basada en fonética perceptiva. Se explicará sucintamente el tema a continuación.

Distancia Fonética

Esa medida es precisamente una idealización de la métrica de la percepción humana, pues pretende medir cómo un humano percibe la distancia entre palabras desde un punto de vista fonético.

En esta métrica se propone una distancia de 1,0 cuando el cambio entre las palabras sea de un solo fonema "intenso" y muy diferente. Éste número será menor si ambos fonemas son similares y tanto mayor cuanto más diferentes 'suenen'.

La teoría de esta métrica y el algoritmo de cálculo, fueron desarrollados y publicados en el 2007, fundamentado con una experimentación sobre 100 personas. Los resultados están en el trabajo [16] y lo que resulta interesante, es que esta métrica resulta útil para determinar la percepción 'humana' de diferencia o similitud entre dos formas escritas. En la **sección 8** cuando nos enfrentemos a la necesidad de una restauración multiletra volveremos sobre este tipo de algoritmos que resultan de suma utilidad en esa tarea.

La similitud fonética es intuitivamente pasible de ser usada para evaluar la similitud entre palabras corregidas, mas tiene un problema importante: puesto que cuando una

palabra está con errores del tipo tipográfico, la pronunciación fonética de esa palabra errada, puede ser defectuosa y/o inexistente o impracticable, por ejemplo: ‘tmporal’

Este inconveniente proviene de que las 'reglas' de conversión de palabras a fonemas solo funcionan y están definidas para secuencias de letras válidas de cada idioma y un error de ortografía cambia el panorama y puede no comulgar con las reglas; resultando en un fonema desconocido o no-representable. Este hecho empeora notablemente la calidad del cálculo de la similitud fonética. En la práctica, apenas un puñado de transiciones entre letras está contemplado en los algoritmos [34] que definen la transcripción fonética de grafema a fonema.

No hallamos manera elegante ni práctica de sobrellevar esto, por lo cual esta similitud se usa como una métrica tentativa de calidad o verosimilitud, cuando el error se presume del tipo fonético, lo cual explicaremos a continuación.

5.8 Métricas para Errores en Palabras

Para hallar y clasificar los errores en las palabras, definiremos una métrica apta para la evaluación de una operación sobre una palabra en un contexto lingüístico y perceptivo, definiendo claramente valores asociados a la calidad de la reconstrucción o estimación de ésta misma junto a la expectativa de bondad de lo que se determinó. Finalmente, en la última sección se enumerarán métricas especiales, aptas para medir clasificadores.

Para esto debemos de introducir brevemente un tema previo, un tanto trivial pero necesario en este punto para poder plantear el aporte de estas métricas.

Teoría Lógica de la Prueba

La filosofía que sustenta a la lógica indica que ésta se utiliza por lo general para denotar alguna propiedad y su álgebra pretende probar algo al respecto, o en su defecto aplicar reglas para deducir algo más, que a '*prima facie*' no se sabe o no es aparente.

Hay numerosos escritos sobre el tema y uno bastante interesante de la escuela de filosofía de Stanford, se encuentra publicado en internet [43].

Métrica vs. Lógica

La idea que una métrica estime algo sobre una entidad o proceso, es una afirmación del tipo puramente lógico y hasta filosófico, en especial cuando se trata de métricas perceptivas, pues una métrica tipo lógica o absoluta, debe afirmar o negar algo, en cambio para decir "*estimo algo*" o ‘*percibo algo*’ se requiere tener en cuenta algo más que la lógica tradicional, la que repasaremos brevemente, antes de plantearlo.

Lógica Tradicional

La lógica es una ciencia en sí misma, derivada de la filosofía y a medio camino con el álgebra; posee un conjunto de preceptos que se usan desde álgebra básica hasta en las funciones más avanzadas de la ingeniería y computación.

La rama de la lógica que se utiliza mayormente en estas ciencias, es la lógica binaria o de *Boole*[20], con dos únicos estados posibles: el ***Falso*** y el ***Verdadero***.

Dentro de esta lógica binaria o de dos estados, hay un álgebra completa desarrollada y muy usada por los electrónicos e informáticos, llamada álgebra de *Boole*.

Este álgebra contiene axiomas o preceptos de completitud en donde algo solo puede ser ***Verdadero*** o ***Falso***, los conceptos son absolutos y el universo es acotado o numerable, el concepto de conjunto prevalece, junto con todas las operaciones entre conjuntos como ser: intersección, unión, diferencia simétrica, etc.

En esta lógica, la pertenencia es unívoca y binaria (*o algo está o no está en un conjunto*).

También cabe destacar que se define la reflexión de la doble negación o lógica negada: es decir ***Falso (Falso)*** es ***Verdadero***, justamente deducible por la condición de pertenencia, conjuntos y universo finito.

Ahora, analizando un poco de gestación de la misma, esta lógica binaria en realidad, es una idealización y clara simplificación de la lógica humana.

Si bien sirve para calcular (*en muchos casos estimar*) probabilidades y es responsable de haber llevado la computación hasta lo que es hoy en día, tiene escasa aplicabilidad cuando se trata de evaluar y modelizar funciones de percepción y cognitivas. Por ende no siendo óptimas para crear modelos y evaluadores parecidos a los de índole humana, sin una fuerte idealización o sesgo.

Esta es una de las tantas razones que han motivado la búsqueda y desarrollo de otros tipos de lógica, como la que veremos a continuación.

Lógica Difusa (Fuzzy Logic)

El ingeniero Lotfi Zadeh (*Azerbaiyán, 1921*) ideó por el año 1956 el concepto de conjuntos difusos, asociado con un preciso conjunto de reglas de álgebra para el cálculo de situaciones que no son ni Falsas ni Verdaderas.

Con el planteo teórico de esta nueva lógica, se revolucionó el funcionamiento de numerosas cuestiones, una de las más afectadas fue su aplicación (*aunque tardía*) al control automático, creando sistemas más estables que los binarios y con mayor capacidad de predicción como controles de temperatura, sistemas de frenado automático, etc.

Variables Perceptivas

Una de las aplicaciones más notables y que traemos al rescate es la capacidad de la lógica difusa para predecir, generar e interpretar variables perceptivas humanas, descriptas por nosotros los humanos como palabras; justamente como lo son el frío y el

[20] http://en.wikipedia.org/wiki/Boolean

calor para el caso de un control automatizado de temperatura. Veamos un ejemplo simple de conversión entre variables lingüísticas utilizando lógica difusa.

Supongamos que hay una variable que mide la temperatura de un ambiente, para esto hay un termómetro el cual entrega un valor de temperatura numérico con 0.1ºC de precisión y se coloca un cartel el cual debe indicar una palabra que diga cuán frío o caliente está el ambiente, pero en palabras. Esto es claramente una discretización de algo cuasi-continuo, lo cual debe tener una salida lingüística, en palabras.

Modelización Difusa

Supongamos que 25 grados es la temperatura considerada agradable, luego debajo de 18 grados podemos decir que es fresco, debajo de 8 que es frío y así sucesivamente, construyamos una tabla de sensaciones con límites:

Tabla 1

50 ºC	mortalmente caluroso
40 ºC	Insoportablemente caluroso
35 ºC	muy caluroso
30 ºC	Caluroso
25 ºC	Agradable
17 ºC	Fresco
12 ºC	muy fresco
8 ºC	Frío
5 ºC	muy frío
0 ºC	Helado
-10 ºC	Congelante
-20 ºC	intolerablemente frío
-50 ºC	mortalmente frío

La lógica difusa, propone que para un valor de temperatura entre 2 valores contiguos, hay una función de pertenencia creciente y decreciente para ambos lados. Esta interpolación se asume por lo general como lineal, salvo que haya una razón de peso que presuma o indique lo contrario.

Acorde a esto, por ejemplo entre 25ºC y 17ºC se pasa de agradable a fresco, si asumimos que la función es lineal, podríamos decir que en la mitad de este rango, que será a los 21ºC habrá un 50% de la gente consultada que diga que está fresco y el otro 50% dirá que está agradable. Es decir que si estamos variando entre 21ºC *(mitad entre fresco y agradable*) y 25ºC (*agradable*), cada vez más gente dirá que está agradable y menos dirá que está fresco, y viceversa para el otro lado. Ésto demuestra que la lógica difusa o *Fuzzy* posee relación con la estadística, pues representa probabilidades.

No pretendo ahondar en el tema innecesariamente, alargando este trabajo con información que está disponible mucho mejor en los libros específicos del tema y/o internet. Solamente se pretende dejar planteada la utilidad de esta lógica difusa en la modelización de la percepción humana y juntamente mostrar que en el presente trabajo, a cada unidad clasificada (etiquetada - palabra o símbolo) se le ha agregado una variable

difusa del tipo lingüística llamada **Verosimilitud** para modelizar lo perceptual-humano: *cuan verosímil es lo que se ha clasificado/etiquetado*[21].

Verosimilitud

Esta métrica fue definida como una variable difusa, englobando una probabilidad y lógica binaria. Ella acompaña cada etiqueta asignada a una palabra por el sistema, se manifiesta como una variable numérica real con un rango de valores de entre +1.0 y -1.0.

Veamos en la Tabla 2, que significa cada valor de verosimilitud, aplicado a una etiqueta lingüística, junto a todas las clasificaciones y valores difusos asociados:

Tabla 2

+1	Significa que se tiene absoluta certeza de que la etiqueta es la correcta. Es decir la etiqueta es correcta 100%.
0..1	Indica la probabilidad que SÍ sea lo que afirma ser.
0.0	Significa que no se posee certeza sobre la etiqueta aplicada, que es solo tentativa, es un valor indicando indeterminación total.
0..-1	Indica la probabilidad de que NO sea lo que afirma ser.
-1	Significa que la etiqueta aplicada es 100% probable que NO sea la correcta, a pesar de haber alguna evidencia de que lo sea.

Esta lógica de verosimilitud parece extraña y en realidad se podría reducir a un valor de probabilidad adicionado a un valor de lógica binaria, pero se mantuvo de este modo por motivos prácticos, los cuales veremos a continuación.

Operaciones Difusas

Operaciones con esta lógica, supongamos que se trata de operar entre dos variables A y B, a las cuales se han asignado con verosimilitud Va y Vb,
¿Cual sería la verosimilitud del conjunto?

Hemos propuesto que este álgebra funcione así:

$$\begin{aligned}
&Va > 0 \;\&\; Vb > 0 \rightarrow Vab = \sqrt{(Va \cdot Vb)} \\
&Va > 0 \;\&\; Vb < 0 \rightarrow Vab = Va - Vb \\
&Va < 0 \;\&\; Vb > 0 \rightarrow Vab = Va - Vb \\
&Va < 0 \;\&\; Vb < 0 \rightarrow Vab = -\sqrt{(Va \cdot Vb)}
\end{aligned}$$

Si bien no hay comprobaciones ni métricas precisas comparativas, el uso de este álgebra, ha arrojado resultados alineados con lo humano-perceptual; en especial cuando es usado para realizar selecciones entre variables en procesos lingüísticos más avanzados en los que se está trabajando para lograr la aplicación del resultado del presente desarrollo para sistemas de diálogo hombre-máquina. Hemos comunicado esta experiencia en un trabajo previo publicado [19].

[21] Nota: *en lingüística la clasificación se suele llamar etiquetado*

5.9 Métricas para Clasificadores

A menudo, definir el rendimiento o la exactitud de un clasificador binario no perfecto presenta cierta dificultad; puesto que el universo de posibilidades no se cubre con dos estados: verdadero y falso; sino que como el clasificador puede equivocarse, habrá cuatro combinaciones posibles. Por ejemplo:

Si usamos *Positivo* y *Negativo* para definir la salida del clasificador con una muestra, y las palabras *Falso* y *Verdadero* para definir si la clasificación fue o no la correcta, aparecen estas 4 combinaciones que presentamos en la Tabla 3:

clasificación ▼	realidad ►	Negativo	Positivo
Negativo		Verdadero Negativo	Falso Negativo
Positivo		Falso Positivo	Verdadero Positivo

Tabla 3

A partir de esto se pueden definir la exactitud de ese clasificador mediante la cifra de mérito para la calidad de su clasificación.

En la práctica esta precisión se obtiene por experimentación con un número importante de elementos a clasificar, de los cuales se sabe previamente su clasificación real, realizada por otros métodos exactos.

Aclaración

En teoría de mediciones este tema se aborda en algunos textos, con la definición de errores del **Tipo I** y del **Tipo II**, y si bien el planteo es similar; no son así las conclusiones ni las métricas definidas.
Se enunciarán las métricas que frecuentemente se utilizan para caracterizar el rendimiento y comparar la calidad; es decir, hacer mediciones de bondad de los clasificadores utilizados en inteligencia artificial y lingüística.

Exactitud

Definición de ***exactitud*** [22] de un Clasificador

$$\text{exactitud} = \frac{\text{nº de elementos correctamente clasificados}}{\text{nº total de muestras clasificadas}}$$

[22] Hemos utilizado la palabra **exactitud** para el término inglés ***accuracy***, que tiene, según el diccionario dos acepciones: exacto y preciso aunque en ciertos caso en estadística se usa solamente como exacto. Lo empleamos de esa manera para desambiguar la traducción, puesto que hay otro término involucrado del inglés ***precision*** → ***precisión*** que se traducen con igual ambigüedad pero que nuevamente en estadística tiene un significado específico.

Reemplazando con los totales (Tabla 3):

$$\text{exactitud} = \frac{\text{nº de verdaderos positivos} + \text{nº de verdaderos negativos}}{\text{nº total de muestras clasificadas}}$$

En otras palabras, este término refleja cuán capaz es de realizar su tarea correctamente, medido en forma porcentual, relacionando cuánto clasificó bien contra el total medido.

Cuando se realizan clasificaciones, hay además algunos otros parámetros de rendimiento específicos que nos ayudan a lidiar con las cosas no tan exactas.

Se utilizan respecto a una sola de ambas clasificaciones, la de interés para el problema. En este caso será el error de ortografía.

Ellas son la precisión (*precision*) y la recuperación (*recall*). Existe además una especial llamada *F-score*, o simplemente *F,* la cual combina ambas mediante una media armónica ponderada, siendo la más común la ponderada al 50% llamada en numerosa literatura *F1*; permitiendo con un solo escalar porcentual, de 0 al 100%, medir la calidad global de un clasificador para comparar comportamientos.

Precisión (del inglés: precision)

Cuando estudiamos en detalle el comportamiento de un sistema clasificador, se requiere de otras medidas que nos resulten más específicas para analizar su comportamiento y estimar el error cometido por éste, por ejemplo ‘que tan bien clasifica las cosas que consideró positivas’ y esto lo llamaremos precisión de la clasificación positiva.

$$\text{precisión} = \frac{\text{nº de verdaderos positivos}}{\text{nº total clasificados como positivos}}$$

En otras palabras mide cuán buena es la clasificación, cuando el sistema ha clasificado positiva una muestra, reformulando sería:

$$\text{precisión} = \frac{\text{nº de verdaderos positivos}}{\text{nº de verdaderos positivos} + \text{nº de falsos positivos}}$$

Esto permite estimar el error o cantidad mal clasificada, del total de elementos clasificados positivos, pero no habla del comportamiento del clasificador en sí, respecto de la muestra clasificada, tal vez no ha clasificado muchos que eran positivos como tales.

Recuperación (del inglés: recall)

Esta medida permite determinar cuán capaz de detectar muestras que son realmente positivas, del inglés ‘*recall*’ y lo consideraremos como la capacidad de recuperación:

$$\text{recall} = \frac{\text{nº de muestras clasificad as como positivas}}{\text{nº total de muestras positivas}}$$

Reformulando con los conteos de la tabla anterior, resulta:

$$\text{recall} = \frac{\text{nº de verdadero s positivo s} + \text{nº de falsos positivos}}{\text{nº verd. positivos} + \text{nº falsos negativos}}$$

Interpretación de las Cifras de Mérito

Hay algunos problemas con ambas cifras de mérito en especial si se citan una sin la otra. Estos errores se malinterpretan fácilmente, lo que se explicará brevemente:

Si un sistema tiene *precision* del 100%, eso por sí solo no indica que sea bueno, apenas dice que los elementos que clasificó positivos, estaban todos bien. Pero nada habla de si hubo muchos positivos que vio como negativos y en definitiva los clasificó mal.

Del mismo modo un sistema con un *recall* del 100% no indica nada, salvo que recupera el total de las muestras viéndolas como positivas, independientemente de que lo sean; conteniendo tal vez un sinnúmero de negativos, mal clasificados.

Por esto es que es necesaria una medida 'balanceada' que sea favorecida con la cifra real de mérito de ambas situaciones y esta es la llamada cifra *F*, usualmente se usa en inglés: *F-score*, y la explicaremos a continuación:

F-Score

Es una media geométrica o armónica balanceada, entre la *precisión* y el *recall* usando un coeficiente de mezclado asimétrico β para realizar la media, se la llama *Fβ* y su fórmula es:

$$F\beta = (1+\beta^2)\frac{\text{presicion} * \text{recall}}{(\beta^2 \cdot \text{presicion}) + \text{recall}}$$

Lo más común es que se balanceé al 50%, resultando en la medida llamada F1 score, cuya fórmula es:

$$F1 = 2 \bullet \frac{\text{presicion} * \text{recall}}{\text{presicion} + \text{recall}}$$

Esta métrica F1, si fuese del 100% indicaría que el clasificador es perfecto, pero si resultase ser del 80%, si bien no sabemos como se comporta con los positivos y los negativos, sabemos que es bastante bueno en promedio. Mientras que si el *F1* resultase del 20% sabemos por cierto que es mucho peor que el del 80%.

Gold Standard

A un conjunto de datos con una clasificación correcta, se lo suele llamar en estadística como: ***gold standard*** [23] y es de suma importancia que su estadística sea coincidente a la del universo que pretende representar, además de tener la cantidad suficiente de datos con la mayor variabilidad posible, representando todas las posibles situaciones. Como veremos luego, en lingüística esto es difícil de conseguir y costoso.

Métricas para Múltiples Clases

Para los sistemas en los que una clasificación sólo puede ser binaria, es decir pertenecer a una clase o no, la clasificación es correcta si la clase predicha es la misma que la verdadera clase.

En cambio para los sistemas que entreguen más de una clase como clasificación, una clasificación se considera correcta, si una de las clases predichas es la misma que la verdadera etiqueta, en este caso solo resta sopesar la posición en la lista de clases.

En estos casos, si las clasificaciones son jerarquizadas se puede definir parámetros de calidad como que clasificó correctamente en la primera (*la más jerarquizada*), en las primeras dos, hasta en las primeras n.

El tratamiento de este tipo de clasificaciones es complejo y responde más a un análisis detallado de cada problema que a una métrica definida previamente.

Sin embargo se utilizan finalmente las métricas tipo 'F1' derivadas del ***recall*** y ***precisión***, presentando el problema de clasificación multi-clase como un problema binario de a pares, por ejemplo se enuncia la cifra de bondad 'F1' de una clase contra todas las demás, agrupadas. Luego se obtiene una evaluación final, promediando todas las cifras 'F1' obtenidas mayormente en forma geométrica, ya que la menor manda cuando se trata de bondades.

Con esto concluye la sección de métricas, las cuales se usarán a lo largo de los próximos capítulos y en las mediciones finales para dar precisiones a los resultados.

---

[23] Ver en http://en.wikipedia.org/wiki/Gold_standard_(test)

6 Detección y Corrección

Veremos en detalle los diferentes procesos necesarios para la detección y la posterior corrección de errores de ortografía. Se especificarán los requisitos o especificaciones, luego se explicará el mecanismo adoptado, basado en estadística y heurística para finalmente describir el algoritmo diseñado.

6.1 Segmentación de texto

Cuando recibe un texto a corregir, el mismo es originalmente una secuencia de caracteres, y por ende la subdivisión en palabras suele estar realizada por símbolos especiales llamados de puntuación, espacios, listas, etc.

Como se presume que hay errores, nada indica que el error no sea la falta o sobra de uno de estos símbolos 'separadores' de palabras. De hecho hay todo un tema de reparación de pseudo-ortografía, o estilo en donde dos espacios seguidos pueden ser vistos como un error de estilo, o dos puntos seguidos, o un punto y una coma. Esto no obedece a regla alguna, más que un conjunto de heurísticas y no se abordará aquí.

Al cortar las palabras usando estos símbolos, las partes resultantes se denominan comúnmente: *segmentos*, *tokens* o *chunks* (*del inglés: trozo*)

El proceso de corte en partes, se denomina con el término: *tokenización*, que es la castellanización de un verbo inventado a partir de la palabra inglesa adoptada *token*.

Los mecanismos por medio de los cuales se realiza esta tarea de *tokenizado* sobre una secuencia de símbolos son complejos y suelen tener ambigüedades, en su gran mayoría solucionables.

Hay numerosos métodos para esto y podríamos clasificarlos en estadísticos [66] y basados en reglas [11]; los primeros son muy buenos pero requieren de un entrenamiento exhaustivo y si bien generalizan, no logran capturar una gran diversidad de formas con exactitud, por eso hemos elegido los segundos, por ser más seguros y simples de implementar.

Lexers y Autómatas

Esas entidades que realizan el troquelado de textos (*llamado tokenización)* se denominan comúnmente *Lexers* (*del ingles Lexing*) y son implementados mediante autómatas determinísticos óptimos [11]. Estos autómatas son por lo general controlados por tablas y la mayoría de las veces son escritos por un software en forma automática, basados en gramáticas regulares y pertenecen a técnicas complejas de la computación: las gramáticas, los parsers y los compiladores.

En la literatura también se los llama *generadores de analizadores léxicos* o *scanners.*

Un ejemplo de programa de este tipo es el ***Lex*** [24] y todos sus derivados, de los cuales utilizamos uno de ellos derivado a partir del ***JFlex*** [25] como explicaremos mas abajo.

Estos autómatas del tipo *lexers* junto a muchos otros sistemas que reconozcan y hagan cortes de secuencias, sin dudas pueden construirse manualmente; pero en esos casos no hay garantía de que sean correctos, mínimos ni eficientes; ya que la cantidad de estados puede crecer enormemente por todas las combinaciones posibles de símbolos y signos que aparecen; por eso fácilmente se puede perder el control de la programación e incurrir en sistemas ineficientes y hasta cometer errores difíciles de detectar.

Generadores de Código

Se han desarrollado herramientas de software que permiten crear estos autómatas '*lexers*' en forma automática y óptima. Se denominan *lexer-generators,* del inglés generadores de analizadores léxicos. Son de la familia de los compiler-compilers, o sea generadores y compiladores de compiladores como lo es el ***YACC***[26].

Estos '*lexer-generators*' se valen de una gramática regular basada en caracteres y reglas y al ejecutarlas, generan código fuente, el cual luego compilado, resulta en un autómata óptimo, construido por este mismo software, que se rige exactamente por las reglas expresadas en esta gramática de caracteres que le dio origen.

La gramática mencionada es del tipo de expresión regular, incluyendo en algunos casos las cláusulas de *Kleene*[27] y anidación contextual para permitir algún tipo de contexto, embebiendo junto a estas reglas algo de código en algún lenguaje de programación.

En nuestro caso utilizamos un esqueleto de un programa de fuente abierta en Java, llamado ***JFlex***, el cual se tradujo a C# que optimizamos.

A este sistema lo llamamos ***CSLex*** y como se dijo anteriormente, fue específicamente construido para lograr crear un *tokenizador*, que corre en ***CSharp*** (bajo ***.net***).

El manual de uso y lenguaje del mismo y su gramática no forman parte de esta tesis, puesto que no es un desarrollo original, sino que es una herramienta convencional adaptada especialmente para crear el autómata en el lenguaje escogido para la implementación.

La documentación de la gramática de caracteres, es muy similar a la del ***JFlex*****,** la cual se halla fácilmente en la web. Hay numerosas implementaciones de estas herramientas, la nuestra fue tan solo una adaptación más del java a ***C#***.

Nota: la ejecución de esta herramienta, para generar el analizador léxico, es muy veloz, con un lapso de algunos milisegundos para gramáticas simples. Sin embargo, debido a la complejidad de la gramática preparada en este trabajo, para capturar los elementos mencionados anteriormente, su ejecución tarda de 4 a 5 minutos, en un procesador de

[24] http://es.wikipedia.org/wiki/Lex_(inform%C3%A1tica)
[25] http://jflex.de/
[26] http://es.wikipedia.org/wiki/Yacc
[27] http://es.wikipedia.org/wiki/Clausura_de_Kleene

escritorio de x86 de 2.8Ghz. La tabla generada es de más de 5000 estados y el proceso de optimización es complejo, por eso la tardanza.

El *tokenizador* resultante, permite incluir código fuente especialmente creado para tal fin el cual genera los objetos de salida (*tokens*) con constructores especiales permitiendo que se escriban numerosas rutinas específicas para reconocer cada tipo de '*token*'

El listado completo y explicado de los diferentes *'tokens'* o etiquetas definidos por nuestra gramática y una parte de reglas gramaticales utilizadas para su creación, está en el ***Anexo 1***

6.1 Algoritmo Propuesto

Observando detenidamente el fenómeno de errores de ortografía a solucionar, haremos una serie de ensayos con los métodos que nos ofrece la ingeniería, haciendo hincapié en aquellos métodos que nos puedan proporcionar la mayor ayuda posible al menor costo computacional, con bajo uso de memoria y en tiempo real; optimizando la relación costo/beneficio y cumpliendo con las premisas establecidas.

Enfocaremos primero el esfuerzo de comprensión, en el fenómeno de la lectura de texto y del análisis de las frecuencias de 'trozos' de textos, llamados: *unigramas, bigramas, trigramas, n-gramas*; sobre grandes volúmenes de texto, a las que llamaremos corpus.

Simultáneamente haremos un estudio de cuáles son las características cognitivas de la percepción de las palabras para poder crear un algoritmo lo más humano posible, pues ésta es la intención última: desarrollar un algoritmo para corregir texto lo más parecido posible a como lo haría un ser humano con capacitación media.

Especificaciones

El algoritmo a diseñar deberá cumplir al menos los siguientes principios básicos:

- Similitud al mecanismo humano de detección y corrección.
- Capacidad de encontrar ambigüedades fonológicas.
- Clasificación del tipo de error considerado en cada caso.
- Presentar un número solicitado de **n** opciones ortográficas (*alófonos o palabras con cambios mínimos*)
- Ponderar y ordenar las opciones mediante una métrica ***Fuzzy,*** que exprese su verosimilitud *(ser lo que dice ser, ante la evidencia de la palabra original).*
- Que la opción más verosímil, y las sucesivas, coincidan, en lo posible, con las sugeridas por un humano.
- No presentar demasiadas opciones ni cosas disparatadas.
- En último caso, intentar hallar coincidencia fonética.
- Ponderar cada resultado con un ranking tipo difuso llamado *Verosimilitud.*
- Bajo costo computacional y mínimo uso de memoria.
- Plataforma de implementación modular basada en objetos y portabilidad.

Implementación del Algoritmo

Se utilizó un lenguaje de programación moderno llamado *C-Sharp* (***C#***), el cual data del año 2000, posee compiladores en varios lenguajes alternativos como visual basic ***VB***, un lenguaje compatible con java, llamado *J-Sharp* **J#**. Al igual que compiladores para lenguajes funcionales como el *F-Sharp* **F#** y cuyos resultados y bibliotecas se ejecutan cómodamente por una máquina virtual MSIL independiente del origen del lenguaje.

La plataforma .NET utilizada a su vez, es de especificación abierta y está implementada en una gran cantidad de plataformas de hardware, incluyendo versiones móviles, algunas de fuente libre como ***Mono***[28] otras para clientes livianos web como ***Silverlight***[29]***,*** habiendo algunas para micro plataformas y embebidos como ***.net*** Micro-Framework[30] *(implementado para Arduino y algunos otros proyectos embebidos).*

Este mecanismo es similar en arquitectura al lenguaje ***Java2*** pero dada la compra de ***Sun*** por parte de ***Oracle***, las políticas de cambios incompatibles que han llevado en los últimos años y algunos problemas de performance de la plataforma, se ha decidido ***C#***.

En las estrategias y algoritmos se ha tenido en cuenta el hecho de que los recursos son limitados. Se usaron técnicas de compresión de afijos descriptas en [7], con estructuras óptimas compactas como árboles de prefijos y ternarios [5], permitiendo de este modo que estos modelos y algoritmos sean compatibles, transportables y usables en una gran variedad de infraestructuras (portátiles, microcontroladores, escritorio, servers, etc.).

---

[28] Mono for .NET http://www.mono-project.com
[29] Microsoft Silverlight http://www.microsoft.com/silverlight/
[30] http://en.wikipedia.org/wiki/.NET_Micro_Framework

6.2 Tratamiento Estadístico

Como todo fenómeno en el cual se podría usar la estadística, se puede pensar que el origen del ruido simbólico asociado que da lugar a los errores de las palabras, es un resultado de características estocásticas y puede proceder de un proceso ergódico.

Analizando las estadísticas de los errores de ortografía en palabras, surge una fuerte evidencia de lo contrario, lo que se probará a lo largo de esta sección.

Además, si se presume que la variabilidad de los elementos del mensaje como los caracteres dentro de una palabra, son estadísticamente coherentes y originados de un mecanismo ergódico, se cometería un grande y profundo error, pues éste no es definitivamente el caso de las palabras, lo que se probará también a continuación.

Precisamente debido a lo intrincado de los procesos de formación de palabras y conforme se presentó en el capítulo 5; es que los de la generación de los errores mismos, son, sin dudas, de difícil tratamiento estadístico.

Del mismo modo, la aparición de palabras y no-palabras son estadísticamente indistinguibles, debido a que comparten mecanismos de formación y lógica subyacente.

Todo esto apunta a que el tratamiento estadístico clásico de las palabras es insuficiente y no es posible probar en forma sencilla, por ser fenómenos combinatorios o ***NP-duros***, que las estadísticas de las formas escritas que son palabras, se distinguen de las que no lo son.

A los sumo se puede hablar de comparaciones de estadísticas de conjuntos de palabras existentes en diferentes idiomas, lo cual ciertamente se usa para estimar los idiomas, como se verá a continuación, pero no puede afirmar nada respecto al idioma de una palabra no perteneciente a ningún idioma, una no-palabra y menos aún de una forma escrita aleatoriamente formada.

Flujo de Letras en un Canal Ruidoso

Si bien no hay un manual con reglas únicas[31] de cómo se deben crear las palabras, cada lenguaje natural posee sus propias reglas de formación morfológica junto a un abultado conjunto de restricciones expresadas mediante reglas de ortografía del idioma. Esto permite afirmar que el proceso de creación si bien puede parecer estocástico, no nace de un proceso ergódico, de hecho esto se observa analizando un flujo de palabras reales.

Justamente, se ha descripto en numerosos trabajos que la secuencia de letras, en cada idioma, no es aleatoria ni caprichosa sino sigue una determinada lógica, propia del mismo. Si bien esto es cierto, no se ha estudiado la secuencia de letras o caracteres de las no-palabras de ese idioma, resultando igualmente lógico, por lo cual la estadística de esta secuencia no es muy útil, ante el total de posibles formas escritas existentes.

[31] http://www.reglasdeortografia.com/

Aparte de algunas reglas muy puntuales tratando con ortografía y fonética [3], esa lógica nunca ha sido descripta en su totalidad como un proceso asociado al flujo de caracteres; hay evidencia de que poseemos una fuerte capacidad tanto para crear palabras como para clasificar, entender y corregir cosas escritas conforme a alguna lógica. Tal vez haya un sustento de esto en la manera que las palabras nos "suenan" en la mente, pudiendo detectar errores en ellas, aún en el caso en que las desconozcamos por completo.

Esta es una razón por la que inferimos fácilmente un idioma por cómo suena, o por el tipo de alfabeto usado y su cadencia, cuando lo conocemos. Aún cuando no conocemos el idioma de una palabra, ciertamente podemos inferir que no se trata de una palabra perteneciente a un idioma conocido y hasta solemos poder adivinar el grupo de idiomas al cual pertenecería lo escuchado o leído; esto es fascinante y digno de ser analizado.

Inferencia de Idioma

Por ejemplo, los humanos, podemos ciertamente inferir el idioma aproximado de un texto, siempre que conozcamos ese idioma. Esto presumimos que ocurre mediante un supuesto mecanismo cognitivo/mental de 'pronunciación-virtual' con clasificación y reinterpretación pseudo-fonética.

Veamos un ejemplo; probemos leer en voz alta, sin respirar y sin espacios, todo seguido el siguiente rejunte de letras:

subenpaguenestrujenbajen → *esto suena a algo en… Alemán*

Mas allá de la gracia de este ejemplo dicho en broma, realmente SÍ suena alemán.

Sin duda esta particularidad de los idiomas, ha sido estudiada y hay numerosos trabajos [21], en los cuales se puede estimar con bastante certeza cuan perteneciente a un idioma determinado es una palabra desconocida, sin consultar en absoluto con un diccionario.

A fines del año 2006, en el *Laboratorio de Estereología y Mecánica Inteligente de la Fi-UBA*, se ha publicado un trabajo [36], donde se presenta un algoritmo de estimación del idioma, logrando un 75% de precisión en la predicción del idioma español sobre una simple palabra, sobre un diccionario de 46 mil palabras, en forma completamente estadística y sin diccionario interno.

El algoritmo se basó en la estadística de segmentos de palabras, o n-gramas, conceptos que veremos brevemente a continuación.

Este algoritmo se ha perfeccionado para su uso en el actual trabajo, llevando las cifras de mérito entre 89 y 99.8%, según el idioma considerado y la longitud de palabra.

Es importante destacar que las mediciones de la ***Precisión*** de estos algoritmos se hace sobre un conjunto que representa una fracción promedio de 10^{-19} del total de posibles formas escritas, en consecuencia las medidas de ***Recall*** estimadas son extremadamente bajas y absolutamente imposibles de medir exhaustivamente por el tamaño del espacio.

Sin embargo la capacidad de estos algoritmos de separar no-palabras de las formas que son verdaderamente impronunciables, es aprovechada para acotar el universo de búsqueda del algoritmo de reparación léxica, como se verá más adelante.

Bigramas, Trigramas y n-Gramas

El trabajo [36] presentado anteriormente sobre el tema, prueba de que hay una evidencia estadística de que los "*pares de letras*", a los que llamaremos **bigrama**[32], se suceden solo con determinada distribución de frecuencia particular en un conjunto grande de palabras propias de un cierto idioma.

Y estas distribuciones estadísticas son diferentes en cada idioma, siendo similares entre idiomas de origen similar como los latinos y muy diferentes entre idiomas de orígenes disímiles tomados de a dos entre los normandos, germanos, sajones, anglos, latinos, griegos, etc.

Veamos inicialmente un ejemplo de como se arman estos segmentos.

Primeramente se inserta un símbolo especial que no ocurra en las palabras, por ejemplo el guión bajo '_' como comodín para indicar principio y final de toda palabra, por lo cual la palabra *maderas* se convierte en los siguientes pares de letras:

*_maderas_ : _m + ma + ad + de + er + ra + as + s_*

La regla es que una palabra de N letras se puede representar como un conjunto secuencial de N+1 bigramas.

Si este conjunto de elementos, llamados bigramas de letras, se extienden a pares de palabras dentro de una frase, se está trabajando ya con modelos de representación vectoriales de palabras. Esto es estudiado con modelos VSM (*Vector Space Model*) [54] siendo muy importantes y útiles tanto en lingüística, como en otras ciencias.

Entre los muchos modelos VSM, hay uno muy usado en la recuperación de información (*IR*) *Information Retrieval*[33] llamado 'bolsa de palabras' o *Bag of Words*[34] *(BOW);* algunos de cuyos conceptos se estarán utilizando en sub-algoritmos de este trabajo.

N-Gramas

Cuando se toma un conjunto importante y completo de palabras de un determinado idioma y se realiza un conteo de cuantas veces en ese conjunto aparece cada par de letras (*bigrama*), nos encontramos con un número importante de pares, más no es muy frecuente que aparezca todo par posible de caracteres.
En realidad son muy pocos los pares de caracteres que se repiten muchas veces, en relación a todos los pares posibles de crear.

---

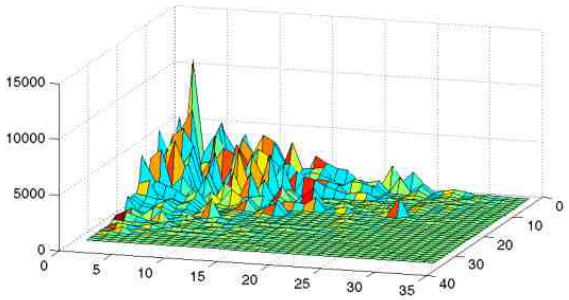
[32] Bigrama: se llama así a un par de símbolos o caracteres de una secuencia, proviene del prefijo *'bi'* : *dos* y el sufijo *'grama'* : *gramaticales*: es decir, relacionado con la posición en una secuencia.
[33] Information Retrieval (IR) ver: http://en.wikipedia.org/wiki/Information_retrieval
[34] Bag of words model http://en.wikipedia.org/wiki/Bag-of-words_model

De esto se puede inferir que toda palabra, cuyos 'bigramas' coincidan con los de mayor frecuencia de aparición en un cierto idioma, es muy probable que pertenezca al mismo.

Por ejemplo si analizamos en el español, para sus 35 caracteres diferentes, el total de combinaciones (*o pares posibles*) es de 35 * 35 = 1225, mientras que haciendo mediciones con listados de palabras en español, los de más frecuencia de aparición son menos del 5%.

Se puede ver en la figura 5, los datos de bigramas de 46 mil palabras españolas clásicas, mayormente del tipo *raíz o lema*[35], en donde se han ordenado de mayor a menor por su frecuencia de aparición, para el total de 35 caracteres mencionados [21].

Fig 5



Frecuencia de pares de letras del español

Se observa que el volumen del paralelepípedo mostrado en la figura, que tiene de base ***35Letras*** x ***35Letras*** y de altura la ***Frecuencia de Bigramas***, está apenas lleno. Esto provee de un espacio o 'aire' estadístico (representado por el volumen no llenado), que sirve para estimar su pertenencia a un idioma. Si todos los pares apareciesen en todos los idiomas con similar frecuencia, la estadística sería neutra y no aportaría evidencia diferenciadora entre idiomas.

Extensa experimentación en numerosos trabajos publicados [33], ha demostrado que la estimación de un idioma mediante estos métodos se logra desde un 60% de precisión hasta un 99%, lo cual es fuertemente dependiente de los siguientes factores:

- Para estimar el idioma de una palabra de menos de 5 letras, la precisión decrece considerablemente; mientras que si se analizan palabras largas o se suma el total de un conjunto de palabras de una frase de más de 30 letras, se obtiene hasta el 99% de precisión.

- La precisión también baja cuando se estima el idioma entre muchos idiomas similares, como ser el español contra todo idioma latino, puesto que poseen muchísimas palabras en común.

[35] Lema o raíz, es la palabra que dio origen a la mencionada por un proceso morfológico de derivación y/o flexión *(en verbos se llama conjugación)*. Por ejemplo **amar** es el lema de: *amado, amando, amé, amaré*; En el caso de adjetivos y sustantivos, se considera lema al masculino singular no flexionado, si existe: **casa** es el lema de *casitas, casona, etc.*

Hay una técnica avanzada, que consiste en utilizar solamente las diferencias de las estadísticas de bigramas entre los idiomas para mejorar el clasificador, junto a otra técnica que utiliza ***trigramas***[36], que aumenta la precisión de la estimación, pero a costa de memoria y tiempo de proceso.

Aquí no hemos usado estas técnicas puesto que la estimación que necesitamos es la de pertenencia al idioma, porque debemos reconstruir una palabra perteneciente al mismo y no nos importan los demás idiomas.

---

[36] Trigrama: es el nombre de un conjunto de tres letras, símbolos o caracteres, proviene del prefijo '*tri*' : *tres* y el sufijo *'grama*' : *gramaticales*: es decir, relacionado con la posición en una secuencia.

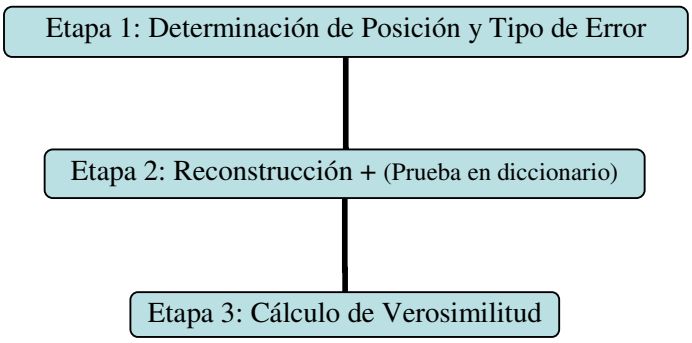
6.2.1 Estrategias para reconstrucción de palabras

En esta sección diseñaremos una estrategia para aplicar a la reconstrucción de palabras y sentaremos las bases de funcionamiento deseadas.

Dado que una palabra es un elemento discreto (*número de posiciones y tipos de símbolos-letras diferentes en cada una*), la elección de cómo proceder a reconstruirla, no es trivial ni al azar. La decisión de por donde comenzar es importante, por lo que iniciaremos analizando cuál sería la posición (*en letras*) en donde comenzar a corregir la palabra, para luego tratar de determinar cuál sería el cambio necesario en esa posición.

El orden de las operaciones es importante, pues alterará la aparición de las opciones obtenidos como opciones válidas. También se debiese crear un algoritmo para determinar cuál es el orden y cuando detenerse en la búsqueda de las posibles opciones para corregir una palabra mal escrita. El gráfico de operaciones sería del siguiente tipo:

Fig. 6



Secuencia de operaciones intrernas

Analizaremos cada caso en particular, para luego poder decidir la estrategia final y construir el algoritmo definitivo.

Se presentan varios métodos que predicen la posición del error y en cada caso, se generará un predictor, luego se los hará competir entre sí. Se seleccionará el método ganador, luego el siguiente y así sucesivamente.

En otras palabras, se determinan todos los predictores ***método+posición*** (etapa 1 y 2; Fig. 6) y se ordenan de mayor a menor por cifra de mérito, en forma sucesiva hasta satisfacer las condiciones del número de predicciones solicitadas.

Finalmente se calcula la verosimilitud (etapa 3) resultante de cada operación exitosa.

Esto garantiza que la reconstrucción se realice siempre con la mayor probabilidad de éxito, independiente del método. Esto último será válido siempre que los predictores sean de calidad.

6.3 Diseño del Algoritmo

En las próximas secciones se presentarán en forma escalonada y detallada las operaciones, divididas en macro-etapas, para definir el algoritmo.

Mayúsculas y Minúsculas

Es importante destacar que en este trabajo no consideramos un error de ortografía a la falta de mayúsculas en un nombre propio o sigla, es decir el sistema es independiente de la capitalización de las palabras, no así de siglas ni de fórmulas químicas o lógico-matemáticas que sí serán reconocidas.

Ubicación del Error

Como las operaciones de edición son focalizadas en "sitios discretos" de las palabras, vamos a tratar de emular y modelizar el mecanismo mental-humano de predecir lugares como "*por dónde pareciera que esta palabra está mal",* lo que es no trivial.

Para lograrlo nos basaremos en la suposición de que el aprendizaje se basa en reglas extractadas de observaciones basadas en las frecuencias de exposición a los datos, que son de naturaleza estadística.

Proponemos que estas estadísticas acumuladas, sean de n-gramas. Éstas deberán ser debidamente tratadas para crear algún tipo de heurística apropiada al problema.

Esto lo veremos en la **Sección 6.3.1**

¿Que letra(s) cambio?

Luego de estimar el o los "*sitios*" del posible error, estimaremos cual de todas las operaciones de edición es más viable para reconstruir la palabra mal escrita, transformándola en la palabra "*estimada*".

Para esto se repasarán algunos métodos que resultaron prometedores y finalmente explicaremos las bases que nos llevaron a la creación del algoritmo. Esto último se verá en detalle en la subsiguiente **Sección 6.3.2**

6.3.1 Estimación de la Posición del Error

Para estimar la posición de un error, se recurrió a la hipótesis que cuando determinada letra ha sido cambiada, reemplazada o insertada, genera una alteración estadística en sus entornos respecto a las frecuencias de las secuencias propias del idioma en cuestión.

Esta alteración resulta ser útil cuando se analizan los errores de ortografía, puesto que una secuencia de caracteres aparece poco o nada en un idioma, en cambio sí aparece en una palabra que supuestamente es del idioma y no está en el diccionario.

En otras palabras, el o los bigramas con menor frecuencia de aparición presumiblemente marquen el 'lugar' en donde está el error.

Se crearon valores basados en heurísticas para la determinación de la posición del error de ortografía y así establecer, en base a la evidencia estadística de frecuencias de bigramas, qué ocurre con estas magnitudes en una palabra, cuando hay diversas operaciones de edición (*rever el tema: Distancia de Edición*). En estos casos aparecen fenómenos interesantes, asociados a las diferentes operaciones de edición mencionadas.

Operaciones de Edición

Analicemos las operaciones intervinientes en la creación de los errores de ortografía. Numerosos autores [73] [75] citan que la mayoría de los errores de ortografía pueden ser vistos como un conjunto de apenas 4 tipos de operaciones de edición, mencionados en la **sección 5.7.1;** a los que se adicionaron ciertas técnicas para tratar sencillamente determinados fenómenos, que suelen aparecer en los textos, citados en el detalle del análisis de la **sección 5.6**.

Cambio/Reemplazo de una letra por otra

`pe`**`r`**`o` → `pe`**`w`**`o`

Es muy probable que las frecuencias relativas a ambos bigramas se hallen afectadas:

```
a la izquierda: 'e'r' → 'e'w'
a la derecha:   'r'o' → 'w'o'
```

Dado que solo el 5 % de los posibles bigramas tienen una alta frecuencia de aparición en un corpus de un dado idioma; un cambio al azar es muy probable que caiga en el restante 95% del espacio de posibilidades, con una nula o muy baja frecuencia de aparición. Un buen estimador podría ser la suma de a pares de estas frecuencias.

Eliminación/Deleción de una letra

`s`**`a`**`ldado` → `sldado`

Al igual que en el caso anterior, al eliminar una letra se forman nuevos bigramas (*las adyacentes a la eliminación*) en este caso: **`sl`**. La frecuencia de este *bigrama* es muy poco probable por el mismo motivo que se mencionó en el caso anterior. Esto permite crear también un estimador mediante la suma salteada de pares creados artificialmente para tal fin (*bigramas estimados*).

Permutación de dos letras contiguas

accionamie**nt**o → accionamie**tn**o

En el caso de una permutación aparecen 3 nuevos bigramas posiblemente fallidos, por lo que un promedio de 3 frecuencias de trigramas puede determinar con cierta certeza la posible posición de una permutación de letras. Esto permite crear un estimador mediante la 'invención' de 3 bigramas cruzados, y su resta respecto de los actuales.

Inserción/Agregado de una letra nueva

invernadero → invernad**e**ero

Es el caso inverso de la eliminación de una letra, por lo cual sucederá lo opuesto, y la posición del posible faltante será la de menor frecuencia de aparición del bigrama. Similarmente a la deleción de una letra, este mecanismo puede crear un estimador detectando la baja frecuencia de los bigramas en forma absoluta.

Corte de Palabras (faltante del espacio)

ho**y_m**ism**o_n**o → ho**ym**ism**on**o

Esta situación es similar al caso de un carácter faltante, solo que todas las secciones de palabras resultantes deberán probarse individualmente en el diccionario.

El estimador usado para predecir los cortes será armado insertando un espacio en el punto en donde la frecuencia de bigramas sea menor, luego se calcula la frecuencia del bigrama creado y si es mayor que el bigrama original, se prueba el corte en ese lugar.

Además esta situación deberá (*y podrá*) tener muchas opciones, dado que un conjunto de *N* letras se puede cortar en *N-1* lugares, pero para cada *N-1* posiciones, habrá *N-2* lugares para realizar otros cortes, pues pueden ser 3 palabras, y así sucesivamente hasta que queden solo palabras de una letra, esto indica que el número de posibles segmentos en que se puede cortar un segmento de *N* letras es *N*(N-1)*(N-2)...* y esto es el factorial *N!* y este número crece peligrosamente rápido.

Este problema del número astronómico de combinaciones dadas por un factorial, constituye uno de los problemas de tipo combinatorio de los llamados **NP duros**[37], los cuales solo se resuelven con heurísticas apropiadas, y suele ser difícil crear métodos que aseguren el hallazgo con éxito de un máximo o mínimo de alguna función planteada en un tiempo razonable, a partir de los resultados, sin explorar todas las posibilidades.

Para tratar de solucionar esto en tiempos razonables, se han limitando directamente el número de intentos de corte, basado en las estadísticas de probabilidad de existencias de

[37] Ver problema ***NP-duro*** en el glosario al final del trabajo.

palabras con diferente número de letras, fundamentado en un trabajo de estadística [38] usando el procesamiento de un corpus español de 450 millones de palabras.

También se han maximizando las longitudes individuales de los segmentos, escogidos con heurística de n-gramas basados en las posiciones de los cortes (inserción de espacios) sólo en donde la frecuencia del bigrama sea mínima y a su vez la frecuencia del nuevo par de bigramas *(luego del corte)* '*espacio_letra derecha*' y '*espacio_letra izquierda*' sean máximos.

Sustitución de Letras múltiples

`hola → hoooollllaaaa`

Cuando suceden estas cosas, se determinó una heurística lógica basada en generar una reducción de todas las letras repetidas hasta el mínimo que predice el idioma, por ejemplo en español la 'L' puede repetirse hasta 2 veces, al igual que la 'R', siendo poco frecuente con las demás letras, por lo que se generarán un conjunto de pares binarios o unarios, y luego de realizará una evaluación de todas las permutaciones de esos pares, escogiendo los de mayor frecuencia total de bigramas para ensayar primero en el diccionario. El predictor utilizado es simple, es una expresión regular que se analiza con un algoritmo recursivo simple.

6.3.2 Estimación de la Reparación a Efectuar

En esta etapa se consultó el estado del arte y se analizaron las diversas herramientas apropiadas y frecuentemente usadas en ingeniería de comunicaciones para la "predicción/estimación de datos" y/o “corrección de errores”.

Se han realizado algunos ensayos teóricos y en los casos en que el método empleado mostraba posibilidades, se realizaron también algunos ensayos prácticos, arribando a la selección de un nuevo método, con la enseñanza aportada.

Análisis de Métodos Existentes

De los numerosos trabajos publicados sobre la utilización en métodos similares, se extrajeron las siguientes conclusiones:

Cadenas Ocultas de Markov

Son conocidas por las siglas ***HMM*** (*Hidden Markov Models*) y se basan en una hipótesis estadística con memoria finita en donde un sistema evoluciona pasando por un número finito de estados y la probabilidad de transición a cada próximo estado, solo

[38] Ver la sección del Cap1. sección *Espacio de Palabras* Fig.2 que muestra la distribución de aparición de palabras por número de letras.

depende de un número finito de estados anteriores y un conjunto de probabilidades de transición desde ese estado.

El mecanismo es muy versátil y permite desde predecir la probabilidad de una secuencia de estados hasta dada una secuencia de observaciones, predecir la secuencia de estados ocultos más probable. Para entrenar estos modelos se usan algoritmos que calculan esas probabilidades de transición, en base solo a las observaciones y al número de estados ocultos estimado, variable bastante difícil de definir correctamente en cada caso.

Cuando se posee la estadística de una palabra, analizando los bigramas anteriores, es posible predecir una letra, prediciendo el bigrama, basado en solamente la probabilidad del *bigrama* anterior. Si bien el algoritmo de *Viterbi* [32] , permite predecir la secuencia de símbolos más probable, en el caso de un error de ortografía, éste no necesariamente es el *bigrama* que repara la palabra, y si no se halla ésta en el diccionario, todo el algoritmo de búsqueda del máximo probable, construyendo el 'trellis', sumado al cálculo de programación dinámica de maximización, serán en vano y habría que re-balancear toda la estadística cada vez que falle, para lograr el próximo valor de menor probabilidad, y así sucesivamente.

Con un simulador simple probamos la efectividad del primer ensayo de *Viterbi*, para palabras con fallas al azar. Se logró apenas el 23% de efectividad en el primer intento, por lo que no pareció atractivo, si bien el concepto es útil, faltaba la calidad para ser usable y el costo/beneficio era demasiado alto.

Estimamos que la falla del método responde a que el entrenamiento no logra capturar las reglas de la estadística que permitan predecir las letras basadas en solamente la letra anterior; el lenguaje tiene demasiadas reglas de mucho más largo alcance (*en letras*). Esto nos indujo a utilizar CRF, una extensión reciente del método HMM.

Conditional Random Fields

Este método llamado abreviadamente: CRF [23], es una evolución de los métodos HMM, y se basa en discriminadores entrenados con secuencias de bigramas, pero a diferencia del HMM reconoce secuencias con más memoria que el precepto de Markov que todo estado solamente es dependiente del último estado y la probabilidad de transición entre éste y el actual estado.

Su nombre proviene de que en lugar de utilizar la probabilidad del último estado observado, utiliza una probabilidad condicional basada en una secuencia de N-estados anteriores y/o posteriores observados, permitiendo una mayor potencia de discriminación pues es equivalente a un HMM de número de macro-secuencias variables, ajustadas cada problema, y allí reside su poder de discriminación.

El método CRF es un potente modelo discriminativo-probabilístico usado hoy en día fundamentalmente en el etiquetado de lenguaje natural y de secuencias biológicas como el ADN. Actualmente es una de las técnicas del estado del arte para este tipo de etiquetado, siendo más poderoso que el método de HMM.

Esta técnica no ha sido ensayada, dada su enorme complejidad. Las bibliotecas que se consiguieron son: *Minorthird* [24], MALLET[25], HCRF CRF y LDCRF [26], CRF++ [27], *CRFSuite* [28] y *Sunita Sarawagi's CRF package* [29]. Ninguna de ellas está portada a C#, lo cual agregaría un trabajo importante de interoperatividad o portación.

Ensayos realizados con los ejemplos de 'demo' con las bibliotecas citadas, usando datos de una variabilidad y dimensión mucho menor al de las palabras de un diccionario, requerían recursos de memoria altos (*un par de gigabytes*) junto a un entrenamiento muy largo, lo cual predice que tal vez exceda a los principios planteados del diseño del presente trabajo, en lo que respecta a recursos limitados (algunos megabytes).

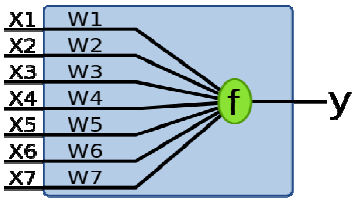
Redes Neuronales

Suponiendo un modelo simple de predicción realizado con redes neuronales artificiales (*ANN Artificial Neural Networks*), se estimó el requisito para construir el sistema y entrenarlo.

Este es el esquema:

Letra a la Izquierda ← *?* *(Letra a Estimar)* *?* → *Letra a la Derecha*

Para resolver el problema de estimación de la letra faltante, se usaría una neurona artificial (perceptrón) para cada letra del abecedario, entrenada con las entradas (codificadas) de N letras a la derecha y N a la izquierda de la letra a predecir o estimar, y luego se entrenarían con todos los pares de letras de cada palabra del diccionario, las salidas competirían por cual sería la que mejor predice la letra correcta.



El esquema muestra un perceptrón, que posee un vector de entrada W (w1, w2,..) el cual es aplicado mediante un producto interno con el vector de entrada X (x1, x2, ..) cuyo resultado es sometido a una función sigmoidea (F) para obtener la salida de predicción, la cual es activada si el sistema predice correctamente la letra.

En este caso de N=1 habrá 70 entradas, formadas una para cada 35 letras a la izquierda y una para cada 35 letras a la derecha. El vector de peso W, en consecuencia será de 70 dimensiones, lo cual no es un inconveniente para el cálculo.

Si escogemos N=2, 3, 4, etc. la complejidad aumenta, sin una garantía clara de obtener resultados mejores, puesto que a mayor distancia en letras la relación es más distante y se corre el riesgo de lo que se llama '*overfitting*' en el entrenamiento de las redes.

También existe otro modelo del esquema, el cual consiste en la posibilidad de usar el número de bigramas a izquierda y derecha, ascendiendo a 2 x 35 x 35 = 2450 entradas, número que lo torna un poco impráctico para el cálculo y convergencia del entrenamiento.

Si bien cada ANN convergerá en el entrenamiento, cosa que está garantizada si hay una estadística no ambigua lo que es equivalente a que el problema sea linealmente separable.

El hecho de que las ANN sean alineales (*por la función sigmoidea interna*), hará que solamente se disparen en ciertos casos una o más neuronas, correspondientes a las letras predichas, lo cual no permite asignar una prioridad en el ensayo de diccionario, tornando el método ambiguo y en consecuencia difícil para optimizar la búsqueda de una solución en secuencia, además de costoso en términos puramente computacionales.

Otro problema subyacente, es que debiesen de entrenarse diferentes grupos de ANN para los principios de la palabra tanto como para los finales, pues hay evidencia fuerte que las estadísticas de bigramas cambian radicalmente con la posición en la palabra, en especial en los extremos.

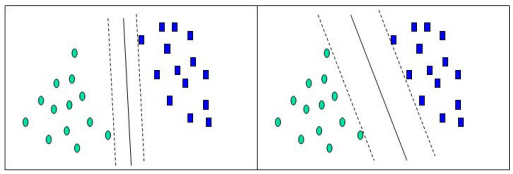
Hay un trabajo del 2003 que propone una técnica de redes ANN [58] usando codificación fonética con distancia de edición de Hamming. Estas redes están diseñadas para inglés por lo que no son aptas para un lenguaje flexivo como el español, cuya estadística es más irregular.

Support Vector Machines

Este método de aprendizaje automático llamado SVN siglas que vienen de *Máquinas con Soporte de Vectores, también llamadas Kernel Machines*, es muy usado en clasificadores estadísticos robustos, procesando fácilmente un gran número de dimensiones de entrada.

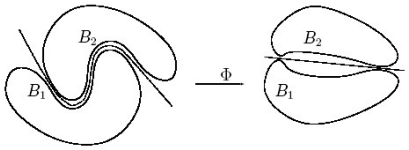
El algoritmo de cálculo y en especial su mecanismo de entrenamiento convergente, fue ideado y propuesto por *Vladimir Vapnik* [22] en 1997, a pesar de que el tema de separadores lineales de máximo margen ya estaba plenamente estudiado desde hacía un par de décadas, pero sin una solución práctica.

El sistema de entrenamiento de la SVN, se basa fundamentalmente en la maximización cuadrática de una función de separación entre conjuntos de muestras a clasificar, como se muestra en la siguiente gráfica muy simple, y donde se ve que el segundo hiperplano, maximiza la función de separación. Los dos hiperplanos son los llamados vectores de soporte, de allí un nombre tan extraño, pues los vectores son los 'soportes' de esos hiperplanos de separación y la máquina a la que se refiere en su nombre, es justamente la función de proyección, llamada *kernel*.



Como se ve en el gráfico, es un método de separación lineal en espacios vectoriales de alto número de variables de entrada (*preferentemente binarias*), que emplea una función generalmente no-lineal (*llamada de kernel*) para proyectar en un espacio de dimensión teóricamente infinita, cuando el kernel es una función no lineal. Y permite resolver allí mediante hiperplanos de separación lineal óptimos con máximo margen, aquellos problemas de clasificación que no tienen solución mediante separación lineal en el espacio natural de los datos, como el caso de las redes neuronales de una capa.

En la figura siguiente, se puede ver un ejemplo mas complejo de la función de Kernel Φ la cual proyecta los espacios B1 y B2 en un nuevo espacio vectorial de dimensión superior, para que sean linealmente separables por un hiperplano.



Si bien está demostrada una equivalencia entre ambos métodos: ANN y SVM, pues ambos maximizan la separación por minimización del error, ninguno de los dos entrega una lista priorizada de las posibles opciones ganadoras, solo se enfocan en la maximización de la separación en el hiperplano, para hallar el elemento predicho con mayor probabilidad de éxito.

Dificultad del uso de ANN y SVM

El inconveniente que se halló, es similar al de las ANN, el sistema SVN predecirá una sola letra luego del entrenamiento y se deberán construir N sistemas predictores para cada letra (*dado el número de caracteres distintos*). Además en caso de que esa letra predicha no sea la correcta, todo el método no tiene segunda vuelta sin un nuevo entrenamiento o una red entrenada anteriormente y un nuevo cálculo.

Esto además, no se podría hacer pues para hacerlo bien, para cada una de las N posibles letras (*35 en español*) se deberá entrenar una nueva versión pensando que la anterior no era la correcta, y así sucesivamente. Lo que da como resultado un sistema de orden combinatorio del tipo ***NP duro***, con un máximo de 35! = 1,03331E+40 clasificadores entrenados para el Español, lo cual es irresoluble y completamente impráctico.

A esto también se suma que en ambos casos las estadísticas de letras difieren ciertamente con cada posición de letra en la palabra, lo cual implica que para hacerlo mejor, se genera la necesidad de multiplicar el número de clasificadores una vez más, por el número total de posiciones posibles en una palabra, que para el español asciende hasta 22 para palabras comunes y 45 para las científicas; tornándolo más inviable aún.

Hay un trabajo [59] que usa una estructura SVM, pero para seleccionar cuáles soluciones alternativas de correctores, como el *ASpell,* son las mejores, por lo que el método propuesto en ese trabajo no aplica como candidato.

Predicción por Máxima Entropía

Este tipo de predictores, también llamados de máxima verosimilitud [80], están basados en la maximización de una función de entropía basada en datos incompletos y al igual que los HMM, SVM y CRF, entregarán simplemente una sola predicción con la máxima verosimilitud, siendo esto impráctico por el costo del entrenamiento y el alto número de clasificadores necesarios para entregar segundas o terceras opciones.

Quantum Analogical Modeling

Una interesante teoría estadística, aplicable a fenómenos lingüísticos fue propuesta por *R. Skousen* en 1989 [77]. Trata de modelización basada en ejemplos, mediante un ingenioso mecanismo de minimización de la inconsistencia en el acuerdo de todos los posibles estados que adoptan las variables en concordancia con los datos disponibles, usando un concepto nuevo llamado contexto y supracontexto.

Lo rescatable del modelo es que utiliza para su evaluación de máximos, el cuadrado del número que indica las frecuencias de las inconsistencias; útil como concepto análogo al de energía, que suele ser el cuadrado de las manifestaciones.

La ventaja del método es que ha tenido éxito en predecir, con muy poca cantidad de ejemplos, fenómenos morfológicos muy complejos en el idioma finlandés[39].

Las desventajas y final limitante fueron que la implementación es compleja y requeriría de computación cuántica, siendo su simulación mediante la generación de todas las combinaciones, dificultosa y poco eficiente.

Boosting

En el año 1988 *Karnes* [45] propuso una hipótesis de método el cual plantea, que cuando hay un problema de clasificación complejo y se tienen solamente un número de clasificadores débiles estadísticamente, se podría crear un nuevo clasificador de mejor calidad, que el mejor de ellos, tan solo combinando y ponderando en cascada esos clasificadores de menor calidad (*llamados blandos o débiles*). Esto se llama *boosting*[40].

Ada Boost

El nombre de este método es una abreviatura de *Adaptative-Boosting*. Surgió de la idea de *Kearnes*, cuando aún no se había resuelto el problema de cómo jerarquizar esos clasificadores creando uno mejor por boosting, de manera no supervisada.

[39] http://en.wikipedia.org/wiki/Analogical_modeling
[40] http://en.wikipedia.org/wiki/Boosting_(meta-algorithm)

De la mano del aprendizaje automático, surgió este meta-algoritmo, ideado por *Yoav Freund* y *Robert Schapire*[44], resolviendo los coeficientes y el orden de esa cascada de clasificadores débiles que genera uno más poderoso que el mejor de ellos.

Si bien esto se resuelve clásicamente mediante entrenamiento y aprendizaje automático, con cada clasificador en cascada, analizando la estadística del problema de los errores de ortografía, no poseemos un 'corpus' de errores, sino solamente de palabras bien escritas, además, la cantidad de errores posibles es prácticamente ilimitada.
Tampoco se pueden determinar cuáles errores serán producidos y con cuál frecuencia, a no ser por muy escasas y cortas listas de palabras con los errores de ortografía más frecuentes, halladas en la web.

Sin embargo las valiosas apreciaciones de sus creadores y el ‘meta’ método general de combinar clasificadores de algún modo buscando una forma óptima, llevaron al planteo del algoritmo final de esta tesis, que se verá en la próxima sección.

6.4 Creación del Algoritmo

El principio de ***Boosting*** de la sección anterior, resultó una opción interesante y apropiada por su filosofía general y le dimos un giro de tuerca, creando un híbrido, el cual mantiene la filosofía con un algo más. Este método resulta además realizable con la información que el análisis de errores de texto nos proporciona, que es una heurística de gradientes basada en la estadística de frecuencias de **n-gramas** en corpus.

Analogía de Entrenamiento

Analizando el concepto de frecuencias de bigramas en corpus, éste resulta totalmente similar a un entrenamiento, puesto que las frecuencias son conteos de apariciones, tal como cuando en una red neuronal artificial (*ANN*), se realiza un cálculo completo y se ajustan los coeficientes en un ciclo llamado 'epoc'. Al igual que con los demás métodos, el utilizar las estadísticas de frecuencias de bigramas ya hechas es como trabajar sobre un nuevo algoritmo de entrenamiento, pero con el problema ya medio resuelto.

Estimación de Parámetros

Dado que la densidad de palabras viables y existentes en español a partir de las 5 letras comienza a ser muy escasa o rala, este fenómeno de poca densidad de datos, que llamaremos 'esparsicidad' (*adoptado del inglés*: *sparse*) conlleva serios problemas en su tratamiento estadístico cuando se procesa lenguaje.

Cuando se entrena un modelo estadístico para aprendizaje automático, se tratan de hallar coeficientes a partir de las observaciones de datos numéricos, sus relaciones o lo que requiera el algoritmo.

Desde un punto de vista estadístico se puede suponer que para entrenar un sistema hay que observar un gran número de ejemplos, pero esto no es así en lingüística dada la poca densidad de pruebas y evidencias de algunos fenómenos. Por más que se agrande el corpus y se procesen muchísimas palabras esperando que la probabilidad de aparición de un fenómeno real aumente conforme crezca este tamaño, esto en realidad no ocurre. Siempre habrá una infinidad de fenómenos lingüísticos que no aparecerán.

Este tema ha sido abordado por numerosos autores [76] que tratan sobre el tema en forma específica. Esto obliga a ser muy cuidadosos en la selección de los métodos a utilizar, analizando qué medir para utilizar las estadísticas apropiadamente. Hay numerosos recursos para estudiar esto en sitios como los de NLP de *Stanford* [41].

Resonancia Adaptativa

Evidencias de este tipo de métodos de entrenamiento abreviado, resultan a partir de teorías de *Carpenter & Grossberg*, sobre las redes neuronales resonantes, basadas en la Teoría de Resonancia Adaptativa [53] (*ART Adaptive Resonance Theory*). Estas redes se fundamentan en principios de aprendizaje que recurren a ejemplos directos, siendo estos métodos complementarios a los que usan la teoría de aprendizaje basada en errores

[41] http://nlp.stanford.edu/fsnlp/

como la propagación inversa (*backpropagation*) en las ANN. En estas redes, directamente se inserta el resultado esperado a un nivel más profundo de la red y solo se ajustan los parámetros para que éste valor resulte el mejor aprendido, en lugar de realizar un entrenamiento basado en minimización de errores.

Otra cosa que se incorporó, análogamente a los algoritmos de *Viterbi*, *SVM* y otros, que hacen uso de la maximización o navegación por gradiente usando programación dinámica mediante funciones como **`argmax(...)`**, es el hecho de hacer competir entre sí cada uno de algoritmos clasificadores, creando una secuencia única de algoritmos aplicables para cada palabra, ordenados en una lista de preferencias.

Esto es similar a la lógica de la programación dinámica: en lugar de ir por gradiente de probabilidades, basadas en los casos anteriores como en el *trellis* de *Viterbi*, el algoritmo diseñado avanza por el gradiente de *'mayores frecuencias'* de N-gramas, haciendo competir los algoritmos entre sí, normalizados luego mediante un coeficiente. Estos coeficientes, son comparables a los coeficientes aprendidos del *Ada-Boosting*

Tomando ideas análogas a la energía y su relación con lo perceptivo [77] también se han utilizado potencias de ciertas frecuencias y parámetros, como ponderadores dentro de los algoritmos, en espacial para predecir el número de búsquedas limitantes, calcular similitudes basadas en parámetros difusos de verosimilitud, etc.

Antes de entrar de lleno en el algoritmo, como este predice palabras que debiesen pertenecer al lenguaje y que sean las más probablemente inferidas por una persona, vamos a calcular la relación estadística entre el número de palabras del lenguaje y las que no pertenecen al mismo, usando como variable el número de letras totales de una palabra, que si bien es una aproximación válida inicialmente, no lo es si en la operación de edición hay eliminación o agregado de letras, alterando su número total.

Esto nos permitirá estimar con suficiente aproximación la dimensión del espacio de palabras posible de crearse y poder crear una heurística que nos indique cuándo detenerse en la búsqueda de nuevas palabras, basada en la relación entre la dimensión de los espacios de soluciones posibles como palabras existentes y la de los errores

6.5 Dimensión del Espacio de Soluciones

Para estimar el tamaño y distribución de datos en el espacio total de exploración de de formas escritas, se realizaron diferentes cálculos para determinar los límites del mismo y cómo están poblados por las palabras reales existentes.

Análisis de Corpus

Con el objeto de estudiar la distribución de estas formas escritas, se decidió analizar un corpus importante. Simultáneamente se determinaron la forma y llenado del espacio de formas existentes. El corpus analizado, llamado LIFCACH2[37], provee un extracto del primer millón de formas escritas diferentes de un total de 450 millones.

Es importante destacar aquí la diferencia sustancial de formas escrita y palabra, dado que dependiente del contexto pueden ser indistinguibles.

Llamaremos aquí **palabra** a toda **forma escrita** que responde criterios lingüísticos, como estar en algún diccionario, ser pronunciable, pudiendo o no tener significado. Ver el glosario al final de este trabajo.

Este corpus provee una lista de formas escritas, presentado a todas como palabras y clasificadas con etiquetas POS, estando muchas de ellas mal escritas y por ende no siendo palabras sino formas escritas.

Se calculó la distribución de la frecuencia relativa de todas las **formas escritas** diferentes, en función de su longitud en caracteres.

Distribución de Palabras

A continuación se muestran los resultados del proceso de cálculo[42] para un número total de palabras diferentes de 89.458, graficando a la vez comparativamente la frecuencia de palabras de un diccionario bastante flexivo, de María Moliner, de 69 mil palabras.

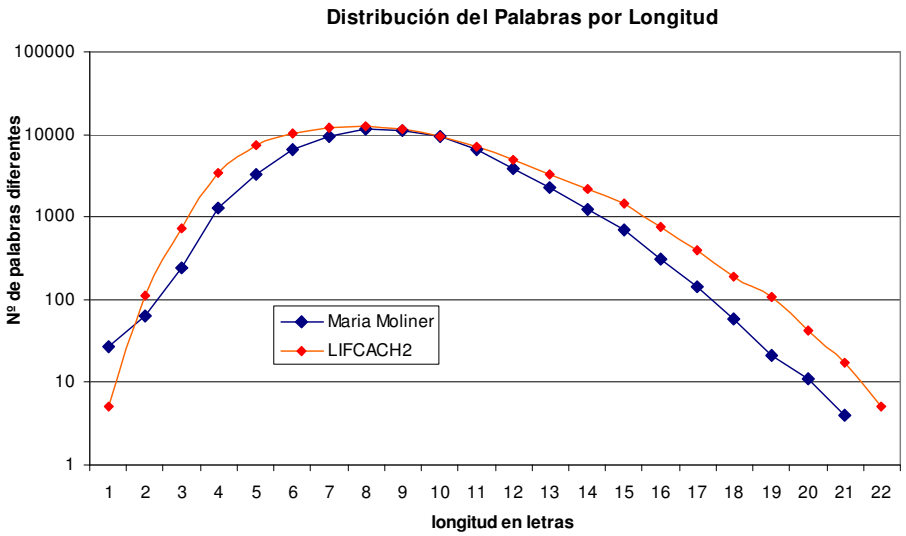


Fig. 7

Los gráficos de Fig.7 se muestran en escala logarítmica de frecuencias, dado el rango dinámico. Esto permitiría estimar rápidamente por la pendiente de las curvas en los diversos sectores, el grado exponencial de éstas, convalidando las leyes de Zipf [43].

Sesgo hacia la Derecha

El corrimiento hacia la derecha de la curva LIFCACH2, medido como coeficiente de variación de Pearson[44] citado en [83], es debido a que las flexiones y derivaciones agregan sufijos y prefijos, en consecuencia las frecuencias de las palabras más largas se ven acrecentadas.

Las formas de estas distribuciones y sus desviaciones han sido estudiadas en profundidad y se condicen con los resultados de los trabajos recientes de *A. Frías Delgado* [83] para el español, mostrando que son una característica propia de la lengua.

Sin embargo, pese a la gran diferencia entre el origen, ambas distribuciones de frecuencias son muy similares en forma, en consecuencia podemos inferir sin cometer mucho error que ésta es la probable '*forma*' de las distribuciones de número de palabras diferentes en función de la longitud en letras de la misma, en un texto cualquiera de ese idioma.

---

[42] En este conteo no se hizo diferencia entre la clase gramatical de la palabra, como ser si era un adjetivo, un verbo, adverbio, pronombre, etc.

[43] http://es.wikipedia.org/wiki/Ley_de_Zipf

[44] http://es.wikipedia.org/wiki/Coeficiente_de_variaci%C3%B3n

Estimación de Dimensión

El espacio de formas escritas posibles de **N** letras con **L** letras posibles es el número de combinaciones con repetición de **L** letras en **N** posiciones y se calcula como $\mathbf{L^N}$

$\mathbf{35^{22} = 9.32 \times 10^{33}}$ palabras diferentes de 22 letras con 35 letras diferentes.

Se piensa que hay también palabras de 21, 20, 19,… 1 letras; la suma resulta:

$\mathbf{9.59 \times 10^{33}}$ palabras diferentes de hasta 22 letras con **35** letras diferentes.

Esto es minimizando las hipótesis puesto que en realidad cuando llega un texto con N caracteres de largo, el número de combinaciones posibles es muchísimo mayor, si incluimos mayúsculas y minúsculas, dígitos y algunos caracteres especiales.

Si hacemos este cálculo en este espacio, un poco más completo, daría del orden de 100 diferentes caracteres, es decir el número será del orden de $\mathbf{100^{22}}$ que es aprox. $\mathbf{10^{44}}$

Si ponderamos esto con las frecuencias de palabras existentes del español chileno [37] el número total de palabras existentes de hasta 22 letras resulta: $\mathbf{4.40 \times 10^{25}}$

Mientras que analizando un diccionario español muy extenso, que incluya todas las conjugaciones, flexiones y derivaciones, el número de palabras, es de: $\mathbf{6.40 \times 10^{6}}$

Esto arroja un grado de 'llenado' de espacio de apenas **1** en $\mathbf{10^{19}}$

Si bien este análisis está hecho en la totalidad de la longitud de palabras de hasta **22** letras, en realidad la cifra se podría calcular para cada rango de longitudes de letras y ponderando con la curva de aparición de palabras, concentrándose en el **95%** de las palabras más frecuentes, aún queda un espacio enorme de opciones, del orden de $\mathbf{10^{6}}$

Conclusión

El espacio de palabras inexistentes que podrían ser consideradas palabras con errores ortográficos y no-palabras es *varios órdenes de magnitud más grande que el de palabras existentes.* Si el número de palabras existentes, incluyendo las flexionadas y derivadas asciende a 10^9, el número de cambios promedio aplicados para distancias de edición de hasta 2 unidades, puede ser del orden de 10^3; esto lleva al número total de palabras 'erradas' viables de ser halladas a 10^{12}, un número respetablemente grande.

Además, dada la naturaleza combinatoria del problema, se hace imposible explorar este espacio de posibles palabras en tiempos polinomiales; a la vez que el costo computacional de las búsquedas necesarias en un diccionario tan grande haría que el requisito de tiempo real para corregir texto 'en vivo' o su aplicación en sistemas de diálogo, sean inviables.

En consecuencia, la exploración del espacio *palabras posibles* debiese ser extremadamente cuidadosa y no tiene sentido expandirse más allá de un puñado de búsquedas cerca de la palabra original por dos motivos: existe una altísima probabilidad de no hallar una palabra existente; y de hallarse una, no se sabría si es la correcta o si se debe proseguir la búsqueda. En la medida que se avance con cambios cada vez más improbables, basándose en la evidencia estadística, se estará acercando cada vez más a la creación de palabras muy poco probables, por lo que una limitación del número de tentativas con cada método aparece como bastante razonable.

Esto lleva a necesitar escoger cuidadosamente modelos estadísticos que sean útiles para esta tarea, siendo los más frecuentemente usados en este tipo de cosas, aquellos basados en *N-gramas* y se abordarán en la próxima sección.

---

7 Reconstrucción Léxica

Como se ha dicho anteriormente, se ha decidido implementar un algoritmo especulativo, de índole netamente estadística y con el concepto filosófico de ***Adaptative Boosting***, [40] sin requerir la adaptación e introduciendo conceptos de ***Resonance Theory*** [53].

Esto resultó viable, dada la enorme dispersión y lo ralo o baja densidad de ocupación del espacio de posibilidades del problema atacado o ***esparsicidad*** *(del inglés* ***sparse****)*, siendo reemplazado el entrenamiento necesario por la utilización de estadísticas de ***n-gramas*** [76], realizadas en el espacio de palabras existentes, con ajustes heurísticos en los coeficientes de competencia, para su normalización.

Este método al que podemos llamar de ***boosting-resonante*** es, en consecuencia, un híbrido ***heurístico + estadístico***, el cual crea sus estados '*pseudo-aprendidos'* mediante una técnica de '*copia'* directa de valores derivados de las estadísticas de los datos correctos basados en una lista de palabras reales de corpus+diccionarios.

7.1 Algoritmo Propuesto

El primer paso es pasar la palabra a corregir por una serie de clasificadores heurísticos 'blandos', los cuales permiten corregir un conjunto pequeño de fenómenos muy frecuentes y en caso de fallar, determinar si la palabra es apta para una corrección más profunda.

El algoritmo luego pasa por una serie de correctores simples que no son compatibles con la corrección posterior y que pueden atacar con éxito ciertos fenómenos como las letras repetidas (*multiletras*) y algunas mezclas de letras y números (*construcciones fonéticas*), creando un set o conjunto ponderado de segundo nivel de ataque.

Se calculan posteriormente todos los estimadores de éxito de cada método, conforme la posición de la corrección en la palabra.

Luego se efectúa una selección y se crea una lista de métodos de reconstrucción, cada uno ponderado con la verosimilitud de éxito dada por el estimador. Ésta lista se ordena

por su verosimilitud, para finalmente avanzar, de una en una, con las pruebas en diccionario flexivo del resultado de cada método de reconstrucción.

El proceso continúa hasta que se halle el número de palabras correctas estipulado o se supere el límite de operaciones permitidas.

Este algoritmo es similar a un 'buscador' de máxima verosimilitud, usando ***boosting*** activo (*competencia*), dado que la búsqueda de la corrección se realiza descendiendo por gradiente (*verosimilitud*) y se maximiza en todo momento la probabilidad de acertar la palabra correcta con la mayor probabilidad inicial, decreciendo sucesivamente.

La principal diferencia con un mecanismo tipo Ada-Boost es que se construye un modelo nuevo para cada palabra a corregir, contrariamente a crear un modelo adaptado a un corpus de entrenamiento, aquí el entrenamiento no existe en esta etapa.

Con el objeto de simplificar su mención reiterada, se le ha dado el nombre de ***spellah*** uniendo la palabra inglesa ***spell***, que significa deletrear y las letras ***ah***, iniciales de mi nombre y apellido.

Los Primeros Pasos

Los pasos generales, si bien incluyen mayores detalles, podríamos resumirlos en los siguientes grandes pasos conceptuales:

Paso 1
Inicialmente, dada una palabra se determina el idioma probable y la verosimilitud de que sea pronunciable, para descartar que sea una palabra "basura" o ruido. Simultáneamente se procuran eliminar y detectar fenómenos especiales como letras repetidas, mezcla de letras y números, junto con algunos errores muy comunes de sustitución, propios del idioma.

Paso 2
Mediante estimadores, se calcula la verosimilitud de cada mecanismo de edición/reconstrucción, conforme su posición en la palabra. Esto obtiene una ponderación de cuán bueno puede ser cada método en cada lugar de la palabra.

Paso 3
Luego se hace competir estos valores de bondad entre sí con un coeficiente de ajuste (*similar al de ada-boosting*) y se obtiene finalmente la secuencia de mecanismos y posiciones preferidas, ordenados por probabilidad de éxito decreciente. Finalmente se ensayan en ese preciso orden hasta hallar el número de opciones corregidas, especificadas como parámetro, o se agote el número de operaciones máximas de tanteo permitidas; lo que ocurra primero.

Este algoritmo se ensayó aisladamente dando buenos resultados, con un vocabulario artificial, pero dado que, para determinar su bondad y compararlo con otras propuestas del estado del arte, es necesario ponerlo a prueba con palabras verdaderas de un corpus lo más extenso posible que sea representativo del idioma español.

Para lograr esto y lidiar con algo derivado de la realidad, se decidió ampliar el espectro de capacidad de reconocimiento de no-palabras y otras construcciones morfológicas para evitar confusiones y falsos errores.

La solución adoptada incluyó el corrector como pieza clave, dentro de un ***Analizador Morfológico*** más complejo, que excede el propósito esta tesis, pero que resultó imprescindible para la continuación de este trabajo. La posible aplicación de los resultados del presente trabajo debiesen generar un producto útil en la práctica y de hecho para probarlo en el campo, hubo que anexar otros bloques especiales.

Es por esto que para no recargar este texto, esas ciertas partes agregadas se describen en los anexos. Entre ellas se cuentan complejos analizadores de entidades nombradas **NER** (químicos, números, unidades, monedas, locuciones, etc.).

Implementación

Para poder realizar las evaluaciones y conclusiones, se construyó un programa dedicado con una interfase gráfica, textual, con botones, campos de resultados. Permitiendo ensayar: funciones, entrenamientos de bigramas y trigramas, algoritmos de aprendizaje automáticos, análisis de corpus, etc.

El sistema de evaluación construido corre bajo Windows (*C#, framework .NET 2.0*), trabajando en modo gráfico. Para el análisis de los algoritmos, se utilizó la biblioteca gráfica *ZedGraph* [45] a fin de poder visualizar y comparar todas las heurísticas, viendo como competían entre sí, permitiendo comparar los resultados y tendencias cómodamente para ajustar opcionalmente los parámetros.

---

[45] ZedGraph a free C# class library for drawing 2D Line, Bar & Pie charts.
http://sourceforge.net/projects/zedgraph/

7.2 Estructura Interna del Algoritmo

La estructura de reconstrucción léxica, que constituye el sector principal del algoritmo de corrección estadística, se podría resumir en el siguiente diagrama de bloques mostrado en la Fig 8:

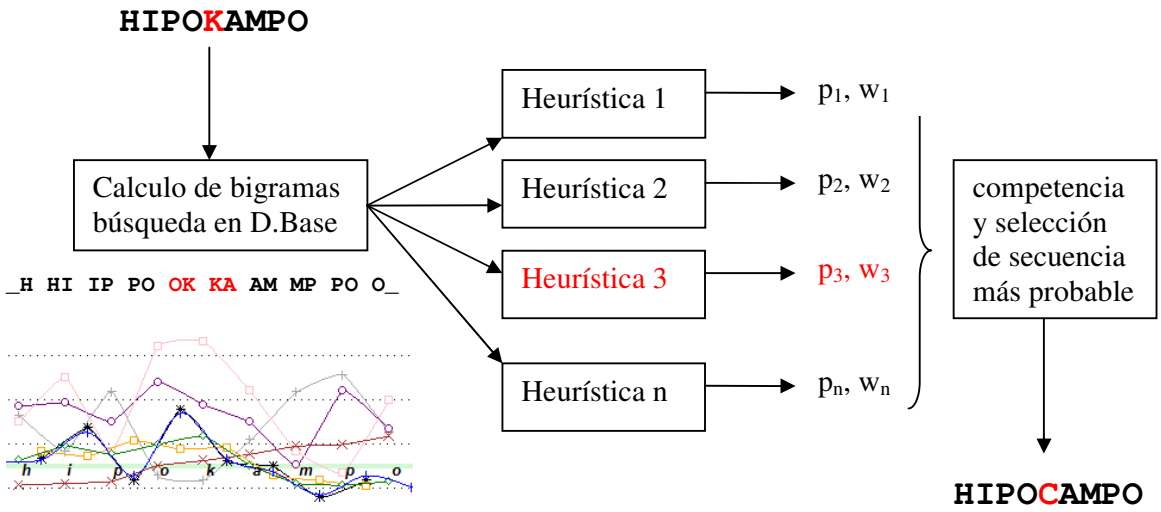


Fig 8

En este diagrama en bloques, se muestra la sección del algoritmo donde a partir de la palabra desconocida **`hipokampo`**, se calculan los 10 bigramas **`_h hi ip po ok ka am mp po o_`** .

A continuación se obtiene para cada *bigrama* su frecuencia en el corpus, usando una base de datos muy veloz (*en memoria*) realizada mediante tablas de dispersión llamadas (*hashtables*).

Luego se calculan todas y cada una de las heurísticas para cada tipo de errores a predecir (*asimilables a clasificadores livianos*) ellos se ponderan finalmente a su salida con un factor de peso de ecualización $\mathbf{w_i}$ y luego compiten entre sí para obtener la secuencia más probable de posición y tipo de operación de edición.

Se puede observar a partir de una corrida real, con la palabra **`hipokampo`**, se interpola de manera de obtener la curva de cada heurística para poder interpretarla mejor. En cada curva que representa los valores más altos: $\mathbf{p_i \cdot w_i}$; valores que significan: '*probar primero*' mediante precisamente '*el mecanismo de corrección especial*', asociado a esa curva, que depende del método/heurística empleada.

La obtención de la secuencia óptima está basada en un simple ordenamiento de las ponderaciones $\mathbf{p_i \cdot w_i}$ de mayor a menor.

Finalmente el sistema ejecuta esa secuencia de operaciones de edición, que estimamos estadísticamente óptima, hasta que se satisface el número solicitado de opciones que son

halladas en el diccionario flexivo, o se agota el número máximo de búsqueda permitidas; lo que ocurra primero.

Ejemplos de Heurísticas

A continuación veremos cómo se desempeña el resultado de cada tipo de algoritmo usando una curva mediante la cual se analizaron las heurísticas creadas, y discutiremos los resultados en detalle para cada una.

Se realizaron ensayos simultáneos a partir de las heurísticas aplicadas a las palabras, cuyos resultados se muestran a continuación con imágenes de las gráficas obtenidas en un programa realizado especialmente.

Se presenta un par de ejemplos, donde se ha insertado la letra **'j'** en el lugar de la **'u'** o **'i'** presumiblemente por un error de posición del teclado ***qwerty***:

`saljdos`

En la figura 9 se observa una serie de curvas, la escala es logarítmica base 2 invertida (más alto es menor frecuencia). Dado que las diferencias entre las frecuencias de aparición son enormes en algunos casos, las curvas presentadas fueron suavizadas con el método de *Bézier* y se observa una clara curva ganadora que es la **diff** (azul, diferencias de frecuencias de bigramas) también la **split** (negro, diferencia salteada de frecuencias de bigramas) luego al **sum2** (verde, suma de a 2 frecuencias de bigramas) luego la **sum3** (naranja, suma de a 3 de frecuencias de bigramas) y finalmente la **swap**, (púrpura, suma cruzada de frecuencias de bigramas).

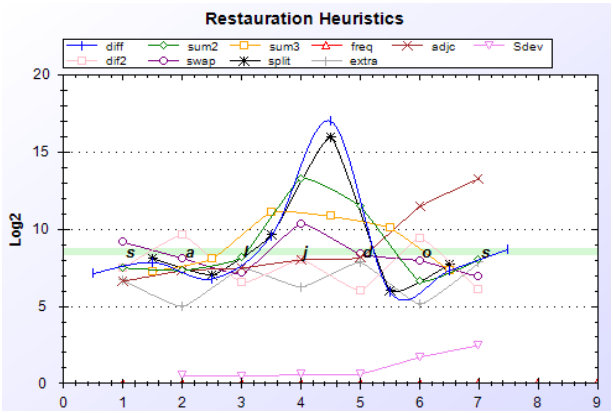


Fig 9

Esto es equivalente a una percepción humana que reflexionaría:

"hmm.. aquí, muy cerca de la 'J', hay algo raro! que no me gusta.."

Luego cada curva, compitiendo con las otras, predice cual es el método a aplicar primero para lograr la máxima probabilidad de éxito en el menor número de iteraciones,

conforme a las reglas del idioma y su frecuencia de letras y bigramas. Los resultados son los siguientes: al solicitarle al menos 2 opciones, el sistema realiza 31 búsquedas tentativas en diccionario y entrega la correspondiente lista de palabras, con el valor de verosimilitud de que sea la correcta al lado, en este caso se puede interpretar como la probabilidad de éxito:

salados : 0.57
saludos : 0.38
salidos : 0.34

De hecho la palabra ***salados*** es mas frecuente que ***saludos***, a la vez que ***salidos*** es un poco menos frecuente, esto permite que el sistema a la vez de hacer un número reducidísimo de búsquedas en diccionario, entregue las palabras en un orden similar al que haría un ser humano común.

Damos otro ejemplo, en el cual se ha colocado adrede la inversión de un par de letras:

sadlos

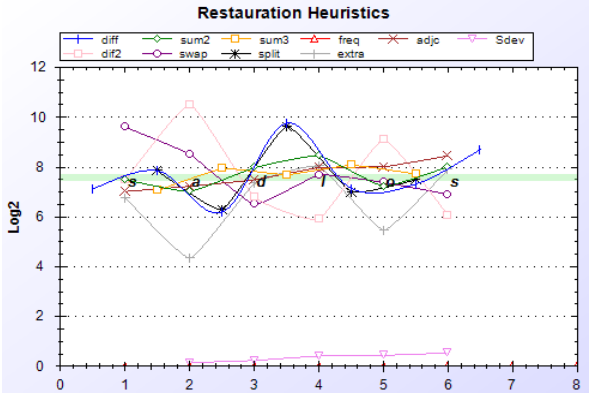


Fig 10

saldos : 0.58

Nótese en la Fig 10, la altura de la curva es menor y sin embargo en este caso el sistema entregó solamente una palabra pero el número de intentos fue grande (251) limitando por el algoritmo por su número operaciones, debido a que no fue hallada una segunda opción con cambios mínimos.

Sin embargo dado el algoritmo se puede ver que prevaleció la lógica humana. La curva **diff** es la ganadora, cuyo método generó la palabra que con alta probabilidad sería correcta: **saldos**, coincidiendo con la palabra desde la cual se partió.

7.2 Validación de Palabras

En esta sección veremos cuantas operaciones de validación en diccionario se debiesen hacer, basándose en la estimación del tipo de operación de edición que se presume ocurrió cuando se introdujo el error.

Número de Operaciones de Edición

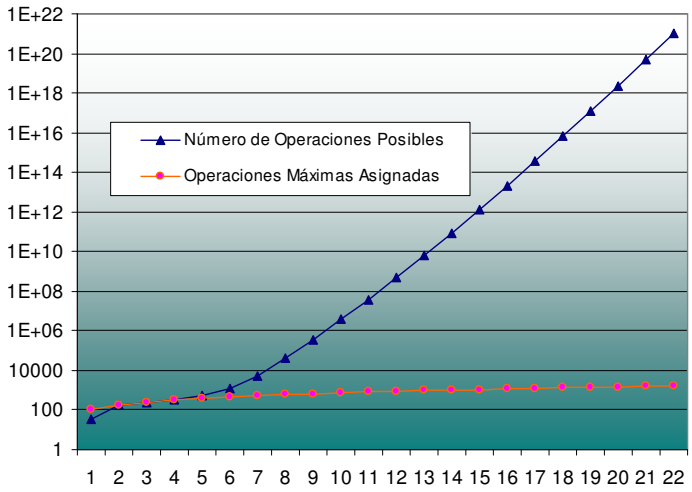
Para esto calculamos cuántas búsquedas en diccionario debiéramos hacer al corregir una palabra, comenzando por restaurar una sola letra de una palabra de ***N letras***, según el algoritmo a aplicar, contando que hay 35 letras diferentes:

```
Eliminación:        N
Inserción:          35*(N+1)
Sustitución:        35*N
Permutación:        (N-1)
Corte en Palabras:  N!
```

Si aplicamos esto para una sola letra de *'reparación'* habría, conforme a la longitud de palabra en letras, los siguientes números de operaciones en la tabla 4 y el gráfico asociado:

Tabla 4

Letras	Operaciones
1	35
2	180
3	256
4	346
5	514
6	1186
7	5578
8	40930
9	363562
10	3629554
11	39917626
12	479002498
13	6227021770
14	8,7178E+10
15	1,3077E+12
16	2,0923E+13
17	3,5569E+14
18	6,4024E+15
19	1,2165E+17
20	2,4329E+18
21	5,1091E+19
22	1,124E+21



Esto nos muestra que la aplicación de una reparación de tan solo una letra, es un problema serio en sí por el número de operaciones de búsqueda en diccionario necesarias para explorar el universo de palabras factibles de generarse, asumiendo que cada búsqueda en diccionario demora un tiempo finito.

Antes de proseguir, y puesto que la comprobación de la existencia de una palabra 'reconstruida' en el diccionario es crucial, vamos a revisar brevemente las especificaciones técnicas que debiera de cumplir un diccionario de este tipo, para satisfacer los requisitos de diseño de este algoritmo.

Tipo de Diccionario Necesario

Para reparar palabras, el diccionario debe contenerlas todas, tanto en raíz como también flexionadas y derivadas, de manera de poder verificar su existencia.

Los diccionarios comunes sólo suelen contener algunas derivaciones, las más frecuentes y poquísimas flexiones; en el caso de los verbos no incluyen todas las conjugaciones.

Para corregir palabras en español, en consecuencia, se requiere de un recurso que no existe ni está disponible fácilmente; de todos modos, para tener una idea de la dificultad de búsqueda de una palabra flexionada o derivada, primero tratemos de calcular, o en todo caso estimar el tamaño de este conjunto exhaustivo de palabras correctamente escritas, que por abuso de lenguaje llamaremos diccionario.

Para solucionar el problema de buscar en este diccionario largo, en forma rápida y con poca memoria operativa, se necesitarán soluciones que contemplen esas restricciones, incluyendo las mecanismos expuestos y otros más que hemos evaluado. Expondremos la solución adoptada en la próxima sección.

7.3 Diccionario Usado

Luego de un trabajo de investigación, análisis y pruebas de un año, se concibió y desarrollo una solución al problema, que cumplía con los requisitos impuestos.

El trabajo resultado de esta investigación fue publicado [17] y el sistema desarrollado se denomina "***diccionario flexivo para el español***" teniendo baja impronta de memoria y siendo suficientemente veloz fue realizado íntegramente en lenguaje C#.

No se presentarán aquí los detalles del algoritmo del diccionario flexivo, y solo a título de referencia describimos en el ***Apéndice 8*** aspectos interesantes de sus métodos y su implementación. Explicaremos solamente algunas mejoras introducidas a fin de poder usarlo, considerando exclusivamente las restricciones del presente trabajo.

Performance Requerida del Diccionario

Como se discutirá en profundidad el la sección final, se puede acelerar el acceso ***si/no*** (más de 100 veces) del diccionario usando tablas de dispersión compactas llamadas filtros de ***Bloom*** [79], armadas con todas las palabras del diccionario expandidas por algoritmos de flexión y derivación.

Estos sistemas clasificadores tipo ***Bloom*** binarios (del tipo ***si/no***) tienen una tasa de precisión de falso negativo del 100%, (*nunca equivocan un negativo*) mientras que la probabilidad de dar un falso positivo es controlable y se puede hacer tan baja como se desee con ciertos parámetros que involucran su eficiencia y el uso de memoria. Por esto, en caso de dar positivo el filtro, se utiliza otro mecanismo de búsqueda tradicional de sufijos/prefijos más lento, o de base de datos.

Como la densidad de palabras existentes en el espacio de búsqueda es tan bajo, la mayoría de las veces la búsqueda dará falso, logrando la velocidad necesaria fácilmente.

Viggo Kann[46] [78], en 1998 aplicó técnicas de corrección usando frecuencias de aparición junto a filtros de ***Bloom*** para el lenguaje sueco, que es muy flexivo como el español, obteniendo una performance de búsqueda de 80 mil palabras/segundo con cache, 10 mil palabras por segundo, logrando corregir 100 palabras por segundo.
La prueba se realizó en una estación de trabajo ***Sun SparcStation***, comparable aprox. con un 40% de velocidad de una PC de escritorio actual (2012/13), su programa se llamaba STAVA, y fue escrito en C, no especificando el compilador usado.

Igualmente el diccionario [17], se ha mejorado más tarde durante el posterior desarrollo de este trabajo, optimizándolo para búsquedas flexivas veloces tipo binarias (si/no)

La velocidad original de búsqueda y lematización alcanzaba apenas 1.500 palabras por segundo y con las modificaciones, basadas en concatenar diferentes tipos de árboles, hemos logrado aumentar esta velocidad a cerca de 100 mil validaciones por segundo[47] con lo cual se satisfacía largamente la especificación correspondiente, sin aumentar significativamente el uso de memoria.

Este diccionario usa actualmente[48] una impronta de memoria estática de alrededor de 35 megabytes para el español, posee un cache dinámico por palabra y otro estático definible; la rutina tarda aprox. 2.5 segundos en inicializarse.

Finalmente no fueron utilizados los filtros de ***Bloom*** puesto que la velocidad obtenida sin su aplicación ya excedía los requisitos, como se verá en los capítulos finales.

7.4 Velocidad del Corrector

El tiempo de hallazgo de una palabra en este diccionario *'prácticamente ilimitado'* es menor a 100 µS en promedio, dependiendo de las especificaciones internas que se deseen establecer, como número máximo de anidaciones a calcular, etc.

Esto finalmente permitiría 'intentar corregir' hasta un máximo de 250 veces cada palabra, logrando una velocidad de análisis con corrección (palabra con error) del orden de hasta 80 palabras por segundo, probado con una lista de 100 mil palabras, todas diferentes, de modo de no activar el sistema de caché, igualmente la velocidad depende del tipo de corrección y las mediciones indicadas son el promedio con los errores más frecuentes.

Adelantando conclusiones que aparecen en detalle en el capítulo 11, el sistema implementado, excedió la velocidad especificada de 200 palabras por minuto, que equivalen a ~3.34 palabras por segundo, puesto que es 20 veces más rápido y esto en el

[46] V. Kann. STAVA's home page, 1998. `http://www.nada.kth.se/stava/`
[47] El sistema midió con un PC de escritorio, corriendo Window 7 Ultimate, de 64 bits en un Pentium x86 2020 de 2.9GHz con 8 Gb de RAM DDR3, con .NET 3.5/4.0 + Visual Studio 2010.
[48] La performance de este sistema es ajustable, sacrificando memoria de proceso estática. Una razonable relación memoria/velocidad se halló disponiendo apenas de entre 30 y 40Mb de RAM.

peor de los casos en donde todas las palabras son diferentes. En un texto normal, dado que hay palabras que se repiten, se activaría el caché aumentando el desempeño logrado del orden de 5 a 10 veces más.

Mediciones

Probando el sistema en un PC de escritorio normal, con texto convencional en donde se repiten (estadísticamente) muchas palabras, se logra analizar en promedio 1000 palabras por segundo, con una tasa de errores del 5% (60.000 palabras por minuto).

La estadística más apropiada, asociada al ingreso de datos en múltiples canales es la de Poison, debido a la independencia entre operadores, permitiendo distribuir casi linealmente la capacidad de procesamiento. Esta velocidad admitiría que un solo CPU de escritorio como el descripto, atendiera alrededor de 500 operadores 'profesionales' ingresando datos simultáneamente a razón de 120 palabras por minuto, sin demoras perceptibles; excediendo ampliamente los objetivos impuestos.

Escalabilidad

La performance obtenida, permite dos cosas importantes, entre otras:

- Si este algoritmo se implementa en un CPU de menor potencia y velocidad de cálculo, como el de un celular o dispositivo portátil, se puede implementar con holgura y aún ser muy veloz, puesto que hay una sola persona ingresando datos.

- Si el algoritmo se instala en un servidor con múltiples núcleos el cual posee una potencia considerablemente mayor a la de un sistema de escritorio (10x a 50x), permitirá que se pueda construir un servicio remoto de corrección de texto, atendiendo un número muy grande (>5000) de clientes livianos remotos y simultáneos, por servidor.

Se concluye que esto satisface y supera ampliamente los requisitos iniciales de diseño.

7.5 Reconstrucción en caso de Errores Complejos

En las secciones precedentes se presentaron un conjunto de algoritmos y heurísticas para reparar errores de 1 letra y de 2 letras, cuando éstas están permutadas. No se abordó el caso en que el error es de más de una letra, o si el mismo posee claras características fonéticas.

Si tomamos una palabra de N letras y aplicamos reparaciones de una letra, luego otra, hasta completar N Letras, no solamente perderemos rápidamente la esencia de esa palabra sino que podremos crear cualquier otra palabra existente con ese número de letras, asumiendo que no modificamos la longitud, haciendo meras sustituciones.

Si en cambio usáramos algoritmos que permitiesen la eliminación o agregado de letras podríamos crear casi cualquier palabra a partir de un número suficiente de operaciones de edición.

Las palabras de cierta longitud, en donde han aparecido errores de múltiples letras, pueden ser reconocidas fácilmente por un humano, siendo el mecanismo predominante la representación 'fonética mental', muy a pesar de que se trate de formas escritas.

Disyuntivas

El número de errores considerando más de una letra es considerable y en consecuencia surge un problema que podríamos enunciar con ciertas preguntas:

- *¿por donde empezar a cambiar/reemplazar/permutar letras?*
- *¿cuál posición/letra por cual otra?*
- *¿que orden de operaciones realizo?*
- *¿que tan parecida es la palabra resultante a la original?*
- *¿cuándo dejamos de probar operaciones de edición correctoras?*

Las respuestas a estas preguntas son inciertas, y tal vez no haya una única respuesta ni se pueda establecer una métrica de las percepciones humanas dado al enorme y vasto espacio de búsqueda planteado, y el relativo escaso número de palabras existentes.

Veremos en las próximas secciones como aproximamos una solución mediante un algoritmo de similitud fonética que viene al rescate.

Antecedentes de Soluciones

En el caso de recurrir a una restauración basada en fonética, numerosos autores [67] han propuesto mecanismos, siendo la mayoría de ellos fuertemente dependientes del idioma y la región.

Para el caso cuando no hay opciones fonéticas ni combinaciones de operaciones de edición que satisfagan, se han diseñado otros mecanismos como, por ejemplo, el corte de palabras en secciones, lo que solventa numerosos problemas.

Veremos en la **sección 8** la reconstrucción de errores complejos en palabras, utilizando el concepto de percepción fonética en el desarrollo de algoritmos.

Algunos mecanismos adicionales se mencionarán en la **sección 9** con detalles mínimos, pues no constituyen más que una implementación 'clásica'.

---

8 Restauración Fonética

Como vimos al final de la **sección 7** un problema muy serio es realizar una predicción o reparación cuando hay más de un error de ortografía, o varias letras cambiadas.

Una manera de poder esbozar ese tipo de reparación es si esta combinación *"suena"* parecida a la palabra que se intentó originalmente escribir, la cual será muy probablemente la palabra entendida por un lector humano.

Como ejemplo analicemos la siguiente palabra:

VAYEMA

Difícilmente sea confundida por un humano, con otra palabra que no sea:

BALLENA

Para hallar cómo los humanos detectan esto, es un tema de ingeniería inversa.

Analicemos las diferencias: la palabra **BALLENA** tiene 7 letras y **VAYEMA** tiene 6 (la correcta tiene 1 letra adicional) luego hay 4 cambios de letras: **V*B Y*L +L M*N**

Tratando de realizar una métrica basada en conteo de cambios, esto indicaría que hubo un 66% de errores de letras en la palabra original o un 57%, si basamos el conteo en la palabra final.

Haremos un breve repaso de acústica, para mostrar sus partes útiles para hallar una solución a este tipo de problemas.

8.1 Representación Acústica

Los sonidos que representan acústicamente una palabra, como se mencionó, constituyen un importante patrón de reconocimiento y por ende de similitud. Esto resulta, en consecuencia, útil para reconstruir una palabra desconocida basada en su fonética.

Es decir, conociendo la fonética o sonidos de una palabra se debiese poder hallar un mecanismo para encontrar la palabra más parecida ‘fonéticamente’ de entre un conjunto dado de palabras existentes o válidas.

Analizaremos algunos fenómenos de la escritura, que ponen de manifiesto la posibilidad de realizar esta operación propuesta, con cierta expectativa de éxito.

Los Sonidos del Silencio

En fonética hay letras mudas, como la letra **H** y si bien la correspondencia fonológica no es unívoca, cuando se precede con una letra **C** en español se forma un sonido diferente

conocido como la ex-letra **CH** que constituye un fonema en sí mismo. Pasan cosas análogas con otras letras y en varios idiomas; esto lo tratamos en forma especial.

Esto genera ambigüedades entre la representación grafémica (*letras*) y la fonémica (*sonidos*) creando múltiples fuentes de errores e induciendo a errores cognitivos.

Ambigüedades Fonéticas: Alófonos

Si bien es posible inferir que letras como la **V** y **B** son similares en el sonido producido, esto no es una regla general de ningún lenguaje, puesto que hay otras letras que no suenan igual, pues dependen fuertemente de la posición y de sus letras vecinas.

Un ejemplo en español es la letra **C** que cuando se coloca antes de una vocal abierta o consonante, se pronuncia como la **K** mientras que si se halla frente a una vocal débil se pronuncia como la letra **S**. La lista de excepciones es larga.

En consecuencia parece ser muy es complejo hacer un conjunto de reglas 'a mano' para convertir una secuencia de letras a fonemas, incluyendo todas las excepciones dependientes del entorno de posición. Se vislumbra éste, un algoritmo difícil de realizar prácticamente.

Si estas reglas se lograsen implementar en un programa, el problema real aún no estaría solucionado, pues al reconstruir una palabra, es necesario saber por cual letra comenzar, y no se sabe si ese cambio de corrección, producirá una modificación radical en como suena la palabra, puesto que repercute en un cambio de los pares con las letras tanto anteriores como posteriores a esa posición. Además de esto, saber por cual letra proseguir, si no hubo éxito en este cambio, es otro tema de cierta complejidad.

Otro problema es que la transformación de letras a fonemas no es ni inyectiva (*único valor para cada letra*) ni sobreyectiva, por lo que *no se podría crear la función inversa*: de fonemas a letras. Esto complicaría aún más la solución, si se tratase de comparar los fonemas y determinar que letras los podrían producir, éste es un problema abierto.

Cambios en Múltiples Letras

Analizándolo de nuevo, el problema es a la vez combinatorio y el cambio de **1..N** letras, conjuntamente con la inserción de una o más letras adicionales, lograría una enorme cantidad de pruebas de palabras nuevas en el '**diccionario**', donde lo más probable es que, como resultado de estos cambios, aparezcan una gran cantidad de palabras existentes en el diccionario.

De este modo surge otro problema adicional: no sabríamos cuándo detenernos con las permutaciones, reemplazos y pruebas, pues no podemos saber cuál es la más viable o 'parecida' al no poseer una métrica de percepción humana de similitud entre formas escritas.

Surgen numerosas dudas con este mecanismo, entre ellas: ¿como reparamos esto y por donde comenzamos, sin realizar una infinidad de operaciones de 'tanteo' o edición (*cambios en la palabra + prueba en el diccionario*) y sin traernos una 'bolsa' de

palabras igualmente probables? Esto teniendo en cuenta que no se dispone de una métrica de similitud basada en la percepción.

En la próxima sección arrojaremos luz sobre mecanismos adecuados.

8.2 Introducción a la Fonética

Existe una rama del área de la lingüística cruzada con la ingeniería acústica y el procesamiento del habla, llamada fonética.

Ella estudia fundamentalmente los fenómenos y características asociados a los sonidos, desde cuando las palabras se generan en el aparato fonoarticulatorio hasta cuando están en el aire como sonido, aislando segmentos a los que llaman ***fonemas*** los cuales se condicen de alguna manera especial y única con las letras escritas, llamadas ***grafemas***.

El Alfabeto IPA

A su vez la fonética ha realizado una categorización de los fonemas o sonidos de las palabras, en un alfabeto nuevo, usando símbolos muy especiales que representan los sonidos pronunciables por humanos. Este alfabeto se llama IPA *(International Phonetic Alphabet)*, el cual según Wikipedia, en su versión del 2005, contiene 107 símbolos de fonemas distintos con 55 modificaciones, de todos los idiomas codificados. Además posee una estructura compleja basada en letras (que indican sonidos “básicos”), diacríticos (que especifican esos sonidos) y suprasegmentales (que indican cualidades tales como velocidad, tono y acentuación).

Con el alfabeto IPA, se ha determinado que cada idioma, región y dialecto humano, posee un subconjunto o grupo frecuente, único y característico de fonemas.

Expertos en el tema, para lograr simplicidad y portabilidad, han codificado los grupos frecuentes de fonemas de cada idioma, mediante letras comunes del alfabeto ASCII.

Esto se hizo debido a que los símbolos del IPA son complejos y no todo aquel que no sea lingüista experimentado, los reconoce con facilidad. Tampoco una computadora común, los podría mostrar en un monitor, si no tiene previamente cargado el 'font' o fuente escalable, para graficarlos.

Finalmente se han creado y estandarizado estos subconjuntos de fonemas del IPA, asociados a cada idioma, región o dialecto, llamados SAMPA *(Speech Assessment Methods Phonetic Alphabet)* todos ellos codificables y expresables mediante simple código ASCII de 7 bits.

Estos alfabetos simplificados SAMPA fueron propuestos y creados a principios de los 80 para los seis principales lenguajes europeos; llegando a la fecha de este escrito, a más de 19 idiomas, incluyendo los de mayor habla en el mundo.

Codificación SAMPA

Para el español ibérico, el SAMPA consta de 29 fonemas, mientras que para el español latinoamericano hay algunas variantes más, en particular, el español rioplatense se diferencia en algunos fonemas, debido a regionalismos fonéticos sudamericanos.

En este trabajo se han adoptado un superconjunto ampliado de 35 fonemas SAMPA, para satisfacer algunas palabras 'importadas' desde otros idiomas y las variaciones debidas al yeísmo, lleísmo y cheísmo del español ibérico, latinoamericano y rioplatense. Los respectivos símbolos figuran descriptos en el **Apéndice 3.**

¿Como se Forma?

En cada idioma, han sido descriptas reglas de conversión entre fonemas y grafemas. Hay numerosos trabajos realizados sobre el tema y a partir de ellos, se pueden diseñar reglas algorítmicas computacionales de conversión de grafemas a fonemas. Un conjunto de estas reglas, que se tuvo acceso para esta modelación, ha sido descripto en los trabajos del *Dr. Ing. Jorge Gurlekian* del LIS / UBA Conicet [34].

El algoritmo de transcripción citado, se ha diseñado codificando estas reglas [34], complementado con la introducción de algunas otras, dada la decisión de incluir reglas de fonética propias de otros idiomas, pero frecuentemente adoptados como palabras ‘importadas’ en países de habla hispana. Tal es el caso de la palabra ‘*boom*’ que es aceptada en español y se transcribe por lo general fonéticamente como *BUM*.

Estas reglas excepcionales, resultan necesarias para el tratamiento de transcripción fonética de combinaciones de letras involucrando ‘w’, ‘k’, y secuencias como ‘sh’, ‘th’, ‘sch’, ‘wo’, ‘oo’ que no son para nada letras ni pares frecuentes en palabras de origen hispano, para las que no hay reglas definidas de conversión fonética, o en todo caso si se aplicasen las clásicas, no se corresponderían con la fonética popularmente aceptada.

La solución finalmente creada, se implementó mediante un autómata finito optimizado, escrito en lenguaje C#, y construido con una herramienta CSLex [35] creada ad-hoc, junto a una compleja gramática regular de transducción de caracteres, logrando una velocidad de conversión de 300 mil palabras por segundo (x86 2.9Ghz), lo que permite incluirlo sin introducir demoras significativas.

8.2 Similitud Fonética

Cuando los humanos pensamos en las palabras, hay evidencia de que la zona del cerebro que se activa , llamada *Área de Brocca* [36], es la misma que cuando las escuchamos, a diferencia que al escucharlas se activa adicionalmente un camino desde el nervio auditivo hacia la zona de reconocimiento.

También hay muchas evidencias en el extenso trabajo del experto en imágenes y neurociencias del habla, el *Dr. Joseph Grodzinsky*, sobre que la percepción visual de un objeto, la relación con su nombre y su reconocimiento abstracto, ilumina/activa una zona adicional de reconocimiento ontológico abstracto.

Lo que podemos decir como inferencia, es que si planteamos la similitud fonética entre palabras, basada en la memoria acústica, hecho que está claramente evidenciado en numerosos tratados de psicolingüística, podríamos hallar una métrica, basada en algún algoritmo, sustentado por experimentos con individuos que corroboren la capacidad de predicción perceptivo-humana de similitud entre palabras.

Esto lo realizamos planteando una modelización físico-acústica de la similitud, junto con información medida de las entidades acústicas o fonemas (*energía, duración, volumen, frecuencia y espectro*). Consistió de un algoritmo que permite determinar una buena métrica de similitud, entre formas escritas, imitando la inferencia de la percepción fonética. El resultado se ensayó con 100 personas y fue publicado [16].

Este algoritmo ha demostrado ser muy similar a la percepción humana y brindar una estimación precisa de cuan similar es una palabra a otra en la 'mente' de las personas.

La ejecución detallada del algoritmo es por cierto compleja, dado que compara las palabras previamente convertidas a fonemas *SAMPA* y busca la óptima alineación entre una representación vectorial de los mismos, en 7 dimensiones, usando un algoritmo de flujo por gradiente con programación dinámica llamado *DTW (Dynamic Time Warp)* [37]. Este algoritmo, usado en ingeniería electrónica, es muy similar al de *Viterbi* para inferencia de secuencias bajo modelos estadísticos, llamados cadenas de *Markov*.

8.4 Índices Fonéticos

Bastante antes de la explosión de los sistemas de cómputo digitales, desde la década de 1960, ya había preocupación sobre la coincidencia fonética en los nombres de personas, motivada por los errores que aparecían en los censos de los Estados Unidos, los que por cierto también motivaron el avance y la creación de la computación, allá por 1890.

Esta búsqueda de soluciones al problema de coincidencia fonética, motivó ideas ingeniosas para crear algoritmos capaces de intentar resolver el tema, o al menos atacar el problema y hallar algún tipo de solución. Expondremos una breve reseña de esto.

Soundex

En 1918 y luego en 1922 *Robert C. Russell y Margaret K. Odell* patentaron un algoritmo para tal fin, el cual llamaron *Soundex* [54]. En el mismo se realizan una serie de transformaciones secuenciales sobre pares de letras, convirtiéndolas en un código alfanumérico que comienza por una letra, codificando con tres números las sucesivas consonantes, luego de eliminar las vocales y letras mudas. Debido a que numerosas palabras de similar pronunciación, al transformarse, resultan en el mismo código, se ha propuesto su uso como índice fonético.

El sistema resulta en la práctica relativamente poco útil puesto que esa similitud es un tanto espuria, no tiene métrica de ningún tipo y está fuertemente atada al idioma inglés y aún en ese idioma, si se buscasen cuantas palabras coinciden con un determinado índice, son más de 50, llegado hasta cerca de 100 y no hay ninguna jerarquía ni orden en la lista resultante que indique algo acerca de la similitud.

Este índice fonético creado por el algoritmo de *Russel*, fue modificado en 1930, para su uso en el Censo Americano y su resultante, fue llamada "*American Soundex*".

Posteriormente, en 1985 el algoritmo sufrió cambios introducidos por *Randy Daitch* y *Gary Mokotoff*, adaptándose a los Nombres Propios de Europa del Este, con la introducción de algunas mejoras, como la codificación no sólo de letras sueltas sino que consideraba algunas secuencias de letras. Como contrapartida, también producía más de un código para cada nombre, complicando su uso.

Hay numerosos trabajos de evaluación [61] y propuestas de mejoras [62], los cuales han puesto en evidencia la necesidad de optimizar, sin apartarse del enfoque fonético.

Metaphone

Casi medio siglo pasó antes de que *Lawrence Phillips*, en 1990 publicara[49] un artículo [55], describiendo un mecanismo más avanzado al ***Soundex***, al que llamó ***Metaphone***. El algoritmo se enfocó en la fonética y no en las letras, a la vez que no truncaba en los primeros cuatro códigos, sino que representaba toda la palabra. El inconveniente era que seguía siendo exclusivamente útil para habla inglesa.

Diez años pasaron hasta que el algoritmo fue una vez más modificado en el año 2000, realizando cambios fundamentales y mejorando al antecesor, llamándose ***Double Metaphone*** el cual provee en ciertas situaciones la entrega de un segundo código alternativo. Hay además una cierta correspondencia en la estructura de estos códigos, en especial con sus prefijos. Esta versión trató de solucionar un conjunto grande de situaciones muy irregulares presentes en numerosos idiomas como el italiano, español, francés, griego, alemán, etc.; generando un algoritmo complejo, con un número enorme de posibles contextos. Por ejemplo la letra 'c' puede ser 'vista' en aprox. 100 contextos diferentes.

En el 2008 hubo un avance menor[50], debido a *Alexander Beider y Stephen Morse*, en el cual se detecta previamente el idioma de la palabra para luego aplicar reglas fonéticas específicas al mismo; además provee una solución a las múltiples codificaciones de caracteres diferentes en los diversos idiomas.

En octubre del 2009, nuevamente *Lawrence Phillips* lanzó un producto comercial, útil para codificar tanto nombres ingleses como no ingleses, pero pronunciados en inglés. Lo llamó ***Metaphone-3*** y está disponible[51] en formato fuente y en modo comercial, para numerosos lenguajes de computación.

Comparativas

El avance y perfeccionamiento de estas técnicas, es interesante y vamos a dar solamente un par de ejemplos numéricos para poder evaluarlas.

---

[49] ***Hanging on the Metaphone***, Lawrence Philips. Computer Language, Vol. 7, No. 12 (December), 1990.
[50] ***Phonetic Matching: A Better Soundex***, a. Beider, S. Morse, disponible en: http://stevemorse.org/phonetics/bmpm2.htm
[51] ***Metaphone-3*** http://aspell.net/metaphone/ y http://www.amorphics.com/

Si se buscaran en la web[52] las palabras "similares" aplicando índices, al apellido ***Washington***, entre los nombres existentes en la *Isla de Ellis*, que es una lista de 25 millones de personas que pasó por la isla entre 1982 y 1924, según cada uno de los siguientes algoritmos tendríamos:

- *American Soundex:* *3900 nombres*
- *Daitch-Mokotoff:* *9 nombres*
- *Metaphone:* *4 nombres*

Lo peor de esta ambigüedad es que el algoritmo no provee una manera de medir cuan parecida fonéticamente es una palabra a otra, una vez obtenida, por lo que la utilidad para restauración es limitada. Seguramente el más apto para un editor sea el *Metaphone*, que provee pocas alternativas, mas no las jerarquiza. Sin embargo este inconveniente se resolvió más adelante, inclusive para el español y será comentado en la próxima sección.

8.5 Indexado Fonético

Nuestro algoritmo de similitud fonética [16] permite calcular un número real que mide precisamente la similitud 'fonético-mental' entre dos palabras escritas y ha sido corroborado con experimentos y responde a la medición aproximada de su percepción.

El algoritmo citado, toma dos formas escritas, las convierte en fonemas SAMPA y luego aplica un algoritmo de time-warping para calcular la alineación óptima basada en un conjunto de variables fonoarticulatorias asociadas a los 35 fonemas SAMPA usados.

El verdadero problema de indexado fonético, se presenta cuando debemos hallar una palabra similar en una larga lista de palabras, de miles o tal vez millones.

Es decir dada una palabra externa desconocida, inexistente en diccionario, se le pide al sistema hallar la palabra existente más similar fonéticamente, de entre el número de palabras contenido, del orden de más de 100.000 palabras.

Si tratamos de hacer esta tarea usando el algoritmo, debiésemos de calcular la distancia o similitud fonética de la palabra dada, contra todas las 100.000 palabras, luego ordenar el resultado y determinar la palabra con la menor distancia o mayor similitud.

Esta tarea de hallazgo, realizada de este modo, demoraría aproximadamente 30 - 40 seg. por cada palabra de alrededor de 6 letras, pues el algoritmo de similitud, puede calcular alrededor de 500 similitudes por segundo en una PC de escritorio y depende fuertemente de las longitudes relativas entre ambas palabras.

Este mecanismo de 'fuerza bruta' impide que se pueda realizar correcciones ortográficas, basadas en fonética y realizadas en tiempo real; por ende no satisface el conjunto de las especificaciones del proyecto. Hubo que desarrollar una solución más eficiente.

---

[52] Stephen P. Morse SF, USA http://www.stevemorse.org/

Soundex y Metaphone resuelven esto mediante una transformación muchos a uno y esta relación por ser asimétrica no biyectiva, aporta sólo ahorro de tiempo de cálculo, a cambio de una indeterminación menor, pero que es de todos modos alta.

8.5.1 Algoritmo de Búsqueda Fonética

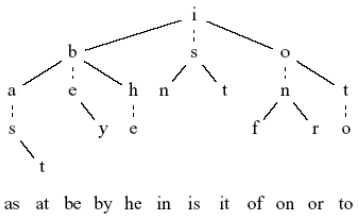
Como es impráctico calcular la distancia de una palabra contra todas las del diccionario, se ha diseñado un algoritmo, basado en árboles ternarios **TST** (*Ternary Search Trees*) [5] previamente explicado en la **sección 7.2**

Estos árboles TST, ligeramente modificados para usar fonemas, si bien permiten una búsqueda que no es tan óptima como una búsqueda binaria, logran un resultado satisfactorio, muy distante de un tiempo lineal, con algunas salvedades.

El mecanismo de búsqueda diseñado es un sistema de descenso por gradiente con acumulación de diferencias, descartando algunas restricciones fonéticas, las cuales luego, una vez hallada la palabra candidata, se calculan mediante el algoritmo de distancia fonética para ver si cumple con las especificaciones de búsqueda.

El mecanismo de navegación en estos árboles se encuentra en [5]. Se explicará la novedad introducida y se provee a continuación una ilustración del mecanismo de búsqueda.

El árbol TST, como se muestra en la siguiente figura, se construye con los símbolos IPA correspondientes a los fonemas en SAMPA. A medida que se construye, si los símbolos son iguales, se crea un descendiente infijo, si la distancia fonética entre los símbolos es menor se crea un descendiente prefijo, y si es mayor uno sufijo.



Cuando se busca fonéticamente, se desea hallar todas las palabras cuya distancia sea menor que un cierto valor o radio de '*distancia fonética máxima*' esperada. El sistema comienza por el primer fonema y se lanza en un mecanismo recursivo sumergiéndose en la estructura del árbol.

El algoritmo luego desciende en este árbol ternario (TST), cuyas 3 hojas contienen caminos alternativos, calculando a medida que avanza, las distancias fonéticas puntuales contra cada uno de los 3 caminos y descendiendo por el de menor valor (*por gradiente*). Esto prosigue mientras la acumulación no exceda el valor de *distancia fonética máxima*, acumulando las distancias parciales entre fonemas a medida que proceso se sumerge en la estructura del árbol, recursivamente, hasta hallar la o las palabras más similares.

Conclusiones y Métricas

El algoritmo fonético presentado no es óptimo ni completo, pero es una aproximación que funciona aceptablemente bien, dado que el problema del cálculo de la distancia fonética completa, excedería las capacidades de proceso y se tornaría rápidamente inviable, por ser un problema NP-duro.

El resultado de búsqueda final logrado, promedia menos de 1 mS, para palabras de 5-8 letras. Por debajo de este tiempo, el lapso del '*tick*' de los hilos de ejecución del S.O. Windows, interfiere en las mediciones y no se tiene buena precisión de medida.

---

9 Mecanismos Adicionales

En virtud a fenómenos de la escritura relacionados con entidades que se expresan mediante secuencias de palabras y símbolos, se incluyen mecanismos heurísticos y de análisis de estos tipos de elementos. Éstos, que pertenecen al lenguaje natural son, por ejemplo, números de tipo ordinal, numeral, cardinal, romano; muchos de ellos expresados con varias palabras.

En esta sección se discutirán, de entre ellas, las más representativas heurísticas utilizadas y se verá cuales son los resultados atribuibles a cada una.

Necesitad de la Limitación del Número de Operaciones

Recordemos que si se prueban todos los cambios aplicables a una palabra de 10 letras, el número de operaciones posibles de edición y prueba, asciende a 3.6 millones de operaciones. Esto es completamente inviable, salvo que tengamos computadoras ilimitadamente poderosas. Pero además, hay otras razones de peso por lo cual no sería una solución.

Si limitásemos a una sola operación de edición para cada una de las posiciones de esta palabra de diez letras, el número de pruebas asciende a 754 cambios simples (inversión, sustitución, supresión e inserción); igualmente esta cifra es superior a la aceptable (100-200), dados los tiempos de búsqueda en un diccionario flexivo y despreciando los demás tiempos de proceso. Debido a esto se generó una tabla que para cada longitud presenta el número máximo de búsquedas en diccionario a realizar antes de abandonar el método de tanteo y pasar al método fonético.

Otra buena razón para utilizar pruebas de tanteo de a lo sumo un carácter/letra, es que la probabilidad de que alguien haya errado dos letras en una misma palabra es de 1/(35*35) o sea menos del 1 por mil. En consecuencia no tiene sentido gastar tiempo en múltiples reparaciones que pueden dar lugar a palabras nuevas que el usuario jamás intentó escribir.

En realidad hay un aspecto más oscuro que se omitió, para estimar las probabilidades y es que en realidad los errores posibles, debiesen incluir mayúsculas y minúsculas, números etc. y entonces se pase de 35 letras a una cifra cercana a 100 letras.

Afortunadamente, hay en los lenguajes cosas que permiten aplicar otras heurísticas para evitar estas multiplicidades, pues las mayúsculas no cambian el significado salvo en contados casos, los acentos (marca diacrítica) se pueden reconstruir con algoritmos basados en reglas de ortografía y acentuación del español, entre otras cosas del estilo, salvo cuando el acento es dependiente de la semántica.

Esto se trata en detalle en la próxima sección.

Mejorando las Estadísticas

Lo interesante es que el análisis de la probabilidad de una falla doble en una misma palabra es mucho menor a **`1/1200`**, tan solo porque un teclado tiene al menos 48 teclas de símbolos y letras en el área central y esto indicaría que es más difícil aún la equivocación doble sin que deje de ser una palabra. Además están las combinaciones de teclas de control como el *shift* y el *control* y el *alt*.

Algoritmo de Limitación

Se ha diseñado un algoritmo matemático simple, el cual calcula cuántas pruebas de diccionario reales se harán basado en la longitud de la palabra, y limitándolo en 250 operaciones. Está basado en la longitud de la palabra y la diversidad total de palabras existentes para cada longitud versus las posibles.

```
MaxDictSeeks = Math.Pow(Length,2) * MaxSug + Length * 15 + 25;
```

El valor **`Length`** es la longitud de la palabra al ingreso al algoritmo, el factor **`MaxSug`** indica cuantas formas escritas como total de análisis y sugerencias deberá clasificar y/o estimar como mínimo el sistema, incluyendo alófonos; su valor es por lo general 1 o 2.

Palabras fuera de diccionario y "ruido"

Cuando se determina el idioma inicial de la palabra, usando el algoritmo descripto en [36] se obtiene adicionalmente un factor llamado verosimilitud, el cual indica cuantos bigramas y trigramas en promedio de esa palabra pertenecieron al idioma analizado respecto a un promedio de las palabras del idioma, normalizando la longitud y frecuencias. Este factor se compara contra un umbral de **`0.05`** y por debajo de esto, es muy probable que la palabra sea netamente de otro idioma, o sea basura.

Palabras Impronunciables

Para detectar si una palabra es completamente impronunciable, se debiese de estudiar si se puede decir, o sea articular esas letras, produciendo fonemas.

Es sencillo entender que hay secuencias de letras no pronunciables. Cuando no hay vocales no hay posibilidad de 'decir' nada parecido a una palabra sin sonidos.

Un ejemplo claro de impronunciabilidad sería: **`mkplhztw`**

Estudiando la pronunciación[53] se logró determinar cuáles son las reglas de formación de todas las sílabas del español, conforme sus componentes como ser: vocales débiles y fuertes, consonantes fricativas, plosivas, labiales, labiodentales, vibrantes, etc.

A partir de allí, creamos un sencillo clasificador como autómata determinístico mediante una expresión regular[54] implementada en la plataforma .NET en forma directa.

---

[53] http://www.uiowa.edu/~acadtech/phonetics/spanish/frameset.html
[54] Expresión Regular o ***Regular Expressions***: http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular

Esto logra rechazar efectivamente palabras que serían impronunciables, con un bajo costo computacional **`O(n+1)`** donde **`n`** es el número de letras de la palabra, debido al tipo de autómata finito determinístico óptimo que produce. El tiempo de proceso de este algoritmo permite cotejar más de 150 mil palabras por segundo, en un PC de escritorio.

Uso del Clasificador

El uso de estas técnicas de clasificación fonético-lógicas, permite tallar óptimamente cuáles tipos de análisis del sistema valen la pena ser usados para cada palabra nueva. Por ejemplo si una palabra prueba ser impronunciable, no tiene sentido buscarla en el diccionario de palabras normales, en cambio sí habría que buscarla en el de abreviaturas, entre los acrónimos, ver si es algo de química o tal vez probar si es un número romano.

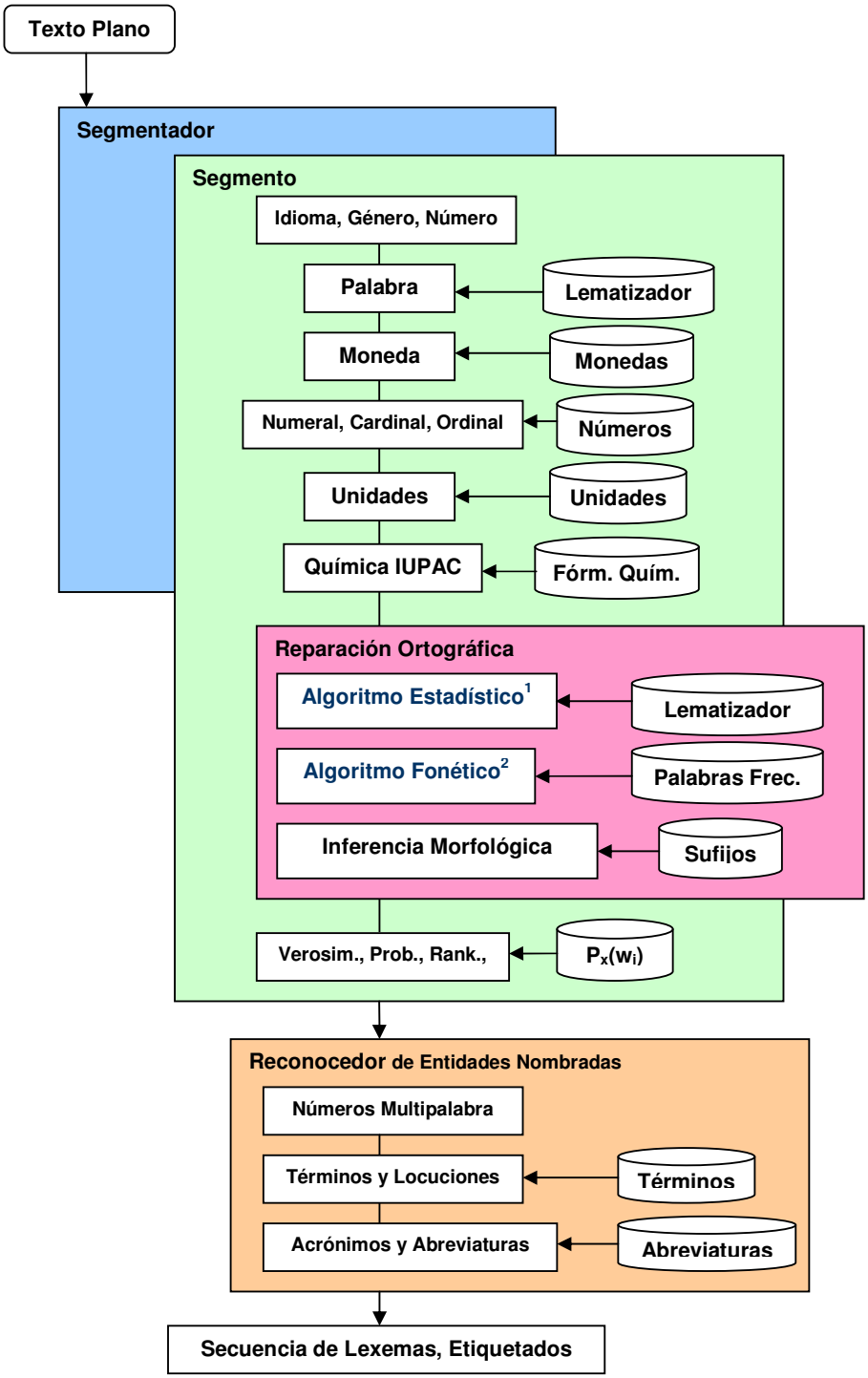
Del mismo modo es conveniente tratar de corregirla ortográficamente, pero la valoración de verosimilitud, por comparación entre la corregida y la original no debiese de ser fonética puesto que una de las dos (*la original*) no es pronunciable; debiendo usarse en estos casos una medida de distancia de edición como la de *Levenshtein*.

---

10 Diagrama en Bloques

Se presenta en la Fig. 11, un diagrama global de bloque del proceso desde que ingresa una secuencia de caracteres o '***string***', hasta la salida de una serie de segmentos clasificados con sus etiquetas lingüísticas. En *Reparación Ortográfica*, se individualizan claramente el **Algoritmo Estadístico**[1] y el **Algoritmo Fonético**[2] descriptos en esta tesis.

Fig. 11



Analizadores Puntuales

En el diagrama en bloques precedente, se han visto un número de secciones que si bien merecen alguna explicación, son de una implementación que no amerita muchos comentarios, siendo fuertemente dependientes de las bases de datos asociadas.
Se explicará en las siguientes secciones, un detalle somero de estas implementaciones.

Palabras

Esta sección se realizó usando el analizador [16] que ya ha sido presentado con anterioridad. Simplemente se analizan las palabras morfológicamente y se etiquetan.

Moneda

Esta sección se realizó usando un diccionario por *hashing* en memoria, a partir de un listado elaborado con los nombres de todas las monedas y sus abreviaturas, siguiendo la norma ISO4127. Es interesante notar que como hay monedas cuyo nombre puede tener más de una palabra, estos términos multipalabra son agregados al analizador de términos múltiples (locuciones o entidades nombradas), de la etapa posterior.

El sistema identificará el nombre de las monedas y su abreviatura, conforme a la norma internacional de nombramiento de estas entidades que es la ISO 4127, entre las que se incluyen las monedas existentes y algunas históricas *como el marco, el franco, el duro y el austral*. Se reconocen tanto los nombres simples "*dólar*", "*peso*", como términos compuestos "dólar canadiense". Esto último lo realiza el módulo de locuciones, al cual se le pasan internamente las palabras compuestas de la norma, junto a su etiqueta ISO.

Unidades

Hay algunas palabras que son las unidades, en los diversos sistemas de medidas, de ingeniería, eléctricos y físicos, que deben tener cierto tratamiento especial. En estos casos hay muchas maneras de decir lo mismo. Es útil, entonces, que estas palabras se etiqueten de manera de ser reconocidas como unidades de medida. En especial si el sistema está concebido para realizar una interpretación inteligente de texto, llamado *Natural Language Understanding (NLU).*

La implementación de la sección de unidades es similar a la mecánica de las abreviaturas, puesto que se resuelve usando un simple diccionario de tabla de dispersión (*hashtable*), garantizando alta velocidad de recuperación con moderado uso de memoria.

El sistema reconoce unos 900 nombres y abreviaturas de las unidades físicas, eléctricas, mecánicas, químicas y otras, en los idiomas inglés y español, tanto del MKS como del CGS y otras unidades inglesas. Simultáneamente desarrollamos un módulo facilitando el cálculo, con conversión y relaciones entre las unidades.

Nombres Propios

Se agregó al diccionario principal morfológico, una lista seleccionada de los nombres propios más frecuentes en español e inglés, así como algunos de origen extranjero de mucha frecuencia de aparición en Latinoamérica.

Para esto se procesaron el padrón electoral argentino del año 2002, junto con las listas de palabras disponibles en internet y aceptadas en los registros civiles de las diferentes provincias y jurisdicciones argentinas y otras de habla hispana.

Para nombres en inglés hay una extensa cantidad de recursos de calidad y simplemente se utilizó una mezcla de varios como el diccionario de *Moby*[55] y listados de grupos de investigación que proveen diversas universidades de Estados Unidos e inglesas.

Hemos incluido también nombres de regiones, provincias y principales lugares geográficos de la zona hispana y algunos muy conocidos del mundo.

La cantidad de nombres propios incluidos es de aprox. 15 mil.

10.1 Inferencia Morfológica

Cuando no ha sido posible clasificar una palabra mediante un conjunto extenso de métodos de reconocimiento, se la considera 'rebelde' y es posible que ‘eso’ sea una palabra desconocida o tal vez una construcción parasintética, al no estar en el diccionario, no podemos afirmar nada al respecto.

Aplicaciones posteriores que usan procesamiento de lenguaje, necesitan saber cual es la posible clase morfológica de esa palabra desconocida, para ello es conveniente estimar qué función puede cumplir gramaticalmente y esto se puede determinar empíricamente y con cierta verosimilitud utilizando diversos mecanismos de estimación morfológica.

Estimación de Clasificación

En español las terminaciones de las palabras suelen estas basadas en ciertas secuencias de letras características que le confieren rasgos gramaticales y semánticos. Estas secciones se llaman sufijos y tienen una relación morfológica con el tipo de palabra y su función, en caso de estar flexionada o derivada.

Utilizando un mecanismo de detección de estos ‘*sufijos*’ se puede inferir bastante bien su clase, en otras palabras: basándose en sufijo máximo, pues puede haber más de un sufijo que contenga otro menor. Por ejemplo sabemos que la mayoría de las palabras plurales terminan en ‘*s*’, este sufijo podría indicar un plural, mas si la palabra fuese ‘*glicó**lisis***’ en realidad posee otro sufijo de índole médica llamado ‘*lisis*’ que indica rotura o destrucción y es de número singular. En consecuencia, al estimar el sufijo más largo el resultado es más preciso, cometiendo menos errores.

Pero como no se sabe cuál es el sufijo máximo ni en donde termina, se realiza una búsqueda mediante un árbol tipo *Trie* de sufijos [39] modificado e invertido. En el

[55] Moby http://icon.shef.ac.uk/Moby/

Apéndice 8 se explica el mecanismo simplificado del *Trie*. Este algoritmo finalmente es muy veloz y provee el sufijo/prefijo más largo coincidente, en el menor tiempo posible.

Una vez obtenido el sufijo más largo, a partir de una tabla construida que se ha elaborado mediante el procesamiento de un extenso corpus del español, se etiqueta la palabra conforme a la estimación, ajustando la verosimilitud con una heurística, combinando la longitud de la palabra, la del sufijo y la frecuencia de aparición de palabras de ese tipo en el corpus.

10.2 Números

Los números poseen un rol importante en la escritura y por eso se hizo hincapié en reconocer nativamente los formatos que consideramos útiles y frecuentes.

Formato Científico

Esta parte del sistema es muy sencilla y su implementación se realiza con expresiones regulares en el *tokenizador* inicial, el cual directamente entrega estas secuencias de caracteres a un '*parser*' nativo en C#, que reconoce tanto enteros como números en formato científico.

Formato Binario, Octal, Hexadecimal, etc.

Este formato requiere únicamente la implementación de un autómata determinístico finito para generar un *parser* simple que al igual que el anterior, se realiza con expresiones regulares y un par de transformaciones, para convertirlo en números del sistema. Los formatos: binario y octal no fueron incluidos en nuestros desarrollos, por no ser de frecuente utilización en textos ingresados por gente común.

Formato Romano

Los números romanos suelen usarse para dar fechas y años. El soporte de números en este formato, al ser construidos con letras, debe estar dentro del reconocedor de palabras, precisamente incluido en el lazo de reconocimiento general en el cual se intenta decodificar este formato antes de entregarlo al reconocedor gramatical. El reconocimiento se resolvió mediante un algoritmo simple que recrea el pensamiento romano, implementado en C#, reconociendo todas las variantes de la escritura de números romanos.

Ordinales, Cardinales, Multiplicativos y Partitivos

Estos números son simplemente una lista de palabras complejas como: uno, dos, dieciséis, primero, segundo, veinticuatro, veintinueveavo, etc.

Fue implementado mediante una lista ordenada de expresiones regulares anidadas de reconocimiento, donde se integró el valor numérico de cada palabra para poder etiquetarlas correctamente con el valor que representan.

Números expresados como palabras

El reconocedor de números expresados como palabras, se construyó mediante un autómata determinístico mínimo, derivado de un listado de estas palabras y usando expresiones regulares. Se contemplaron números cardinales, partitivos o fraccionales, multiplicativos y ordinales. Notemos que solamente se escriben en una sola palabra los números inferiores al treinta.

10.3 Reconocimiento de Entidades Nombradas

A lo largo del trabajo se ha mencionado siempre a la palabra o al símbolo como los segmentos menores cuando se troquelan o segmentan (*tokenizan*) textos.

En realidad existen construcciones ortográficamente aceptadas como 'entidades' por ejemplo las abreviaturas, una fecha-hora, los acrónimos, las locuciones entre otras construcciones multipalabra llamadas términos, que significan una sola cosa, no siendo divisibles por no ser representadas en su significado como la suma de sus partes.

Como la segmentación de estas entidades presenta ambigüedades, es conveniente al realizar la segmentación y clasificación, volver a "*unir*" estas entidades, al menos todas las que contienen símbolos de puntuación, listas, símbolos de matemática, etc.

Esto lo realiza en módulo indicado como "*Reconocedor de Entidades Nombradas*" en el diagrama en bloques mostrado anteriormente, en la Fig. 10.

Se describen a continuación los módulos construidos, junto con algún detalle de implementación que merezca destacarse.

Abreviaturas

Se ha incluido una sección de abreviaturas multisímbolo, las cuales son tratadas con un autómata no determinístico, entregando un reconocimiento intersímbolo, permitiendo incluir puntos opcionales como las abreviaturas de empresas: s.a. y sa.
El autómata realiza la correcta aglutinación en un solo bloque etiquetado, de los símbolos reconocidos por los bloques anteriores, uniendo letras, puntos y comas, y excluyendo eventualmente los espacios.

Fórmulas Químicas

Tratando de llevar este producto terminado al siguiente nivel y dado que una la tarea de todo *analizador morfológico* + *lematizador* + *corrector*, es el reconocer secuencias de caracteres y símbolos que '*quieran decir algo*' cuando nos referimos al ambiente de la química y la bioquímica, hay secuencias de letras muy específicas que representan fórmulas químicas de diversa índole, por ejemplo: H2O es conocido por todos como la fórmula del agua.

La implementación del sistema de reconocimiento de fórmulas químicas, se realizó en dos partes, primero: se implementó un autómata finito determinístico, usando la

herramienta con la que se construyó el ***Lexer*** del sistema analizador, realizado en ***C#*** mediante una gramática como la mostrada en el **Apéndice 1**

La segunda parte en el reconocimiento de estas secuencias, es cuando el autómata de entrada, que oficia de ***tokenizador***, reconoce una secuencia '*probablemente química*', luego la envía a un segmento especial de análisis en donde se construyó un parser recursivo descendente ***LL(k)***, implementando la gramática química de más alto nivel (IUPAC) cuya documentación se adjunta en el **Apéndice 4**

Números expresados con múltiples palabras

Esta sección implementa un sistema que identifica y delimita secuencias de palabras que significan números (*previamente detectadas*) y luego las pasa por un algoritmo de reconocimiento de este tipo de secuencias, el cual se implementa tal cual se forman estas palabras con las reglas de formación de cómo se dicen los números. Esto último es bastante simple y no merece una explicación pues no posee complejidad significativa.

10.4 Parámetros del Analizador Morfológico

El sistema final, en donde ha sido plasmada la reconstrucción de errores de este trabajo, conforma un sistema mucho mayor, llamado ***Analizador Morfológico Robusto***.

Éste posee un conjunto de parámetros que permite manejar las opciones de cada uno de sus componentes; estando formado por 64 *'flags'* lógicos, más un conjunto de elementos discretos lógicos, enumeraciones y numéricos como por ejemplo: el lenguaje presumible del texto, la cantidad de opciones o sugerencias 'diferentes' mínimas a obtener y el número máximo de "*tokens*" a procesar.

El ajuste óptimo de estos parámetros tiene un fuerte impacto sobre la relación de calidad/performance del sistema.

El listado y detalles de cada parámetro, se hallan en el **Apéndice 7.**

10.5 Pruebas y Mediciones

En esta sección se describen las pruebas que se diseñaron junto con las mediciones efectuadas, a fin de hallar los factores de bondad del sistema. Se diseñaron de manera de comprobar el comportamiento de un corrector frente a diversos tipos de textos, contrastando las mediciones con los diversos correctores ortográficos del mercado, que suponemos representan el estado del arte actual.

Planificación de las mediciones

La dimensión del universo de formas escritas posibles, estudiada en la **sección 6.5**, del orden de ~10^{34}, tan sólo para palabras de hasta 22 letras considerando 100 caracteres diferentes *(letras)*.

Si consideramos que los actuales procesadores logran probar aprox. 10^5 "*intentos de reconstrucción*" por segundo, nos quedan, en promedio, aún 10^{29} elementos para comprobar, si se quisiera hacer una prueba sin dejar afuera ninguna opción, esto se llama '*fuerza bruta*' y haría haciendo prácticamente imposible un ensayo exhaustivo, inclusive para probar un pequeño porcentaje de esta cifra, sería necesario un tiempo astronómico.

El verdadero desafío consiste en estudiar una insignificante fracción del universo, asegurando que estas palabras estén cercanas en términos de 'distancia de edición' a las palabras del lenguaje; para que la probabilidad de ser corregida sea máxima. Esto es debido a que más del 90% de los errores de ortografía no están a más de 2 unidades de distancia de la palabra correcta; considerando la distancia de edición de *Levenshtein*.

A pesar de eso el universo de errores, tan solo partiendo de las palabras del lenguaje es enorme. Sin embargo dadas las hipótesis sobre la génesis de esos errores, aquí se estima poder trabajar con un menor número de palabras en relación al universo de las posibles.

Para hacer mediciones reales, dado a que los 'corpus' de contraste, utilizados en los escasos trabajos que tratan el español no están disponibles y muchos de los sistemas publicados tampoco están disponibles para hacer una prueba, decidimos recrear las mediciones con varios conjuntos de palabras, que llamaremos "set de datos", armadas específicamente para este fin.

Primeramente se presentan los criterios para la elección y/o construcción de este *"set de datos"* que fueron usados para las mediciones.

Luego se hablará del criterio tenido en cuenta para la selección de *"correctores"* como exponentes del estado el arte, a los que se tuvo acceso, para realizar las pruebas.

Finalmente se contrastarán y presentarán las mediciones realizadas, para luego discutir los resultados y sacar conclusiones.

Conjuntos de Datos Usados

Se crearon varios conjuntos bajo diversos criterios, contrastando luego los resultados de ellos contra las mediciones aportadas por los trabajos publicados, en caso de compatibilidad. Se obtuvieron algunos conjuntos de datos muy similares, como el listado de palabras de difícil corrección de *wikipedia*, aunque no tenemos una foto exacta de esos listados en la fecha en que se realizaron aquellas mediciones, dado que la fuente de información citada es editada constantemente. Igualmente estimamos que el conjunto que hemos utilizado es muy parecido al set originalmente usado en los trabajos publicados, dado que los resultados de las pruebas con los productos comerciales que hicimos, fueron similares a los publicados.

Errores Artificiales

Si se toma un número grande de palabras del español y se realiza un conjunto importante de operaciones de edición aleatorias de determinado tipo, se obtiene un enorme número de palabras mal escritas artificialmente.

Si bien la estadística de este tipo de errores no es la de los errores cognitivos, puede asemejarse a los de distracción o simplemente a los fortuitos, de índole no conocida.

Usando estos conjuntos, podemos obtener una cifra de mérito del algoritmo correspondiente a cada tipo de edición.

Sin embargo esto no garantiza cómo se comportará el algoritmo en el campo real, pues la distribución y naturaleza de los errores *'artificiales'* no es necesariamente similar a los errores frecuentes o naturales.

La escasa cantidad y la dificultad de hallar los errores reales hace que no se puedan estimar sin sesgo las estadísticas de los mismos considerándolos procesos como de tipo Poisson, según las hipótesis acerca de su origen que se presentaron en la **sección 5.6**

Tampoco sabemos la frecuencia de aparición de los mismos, puesto que hallarlos es muy difícil, su frecuencia es baja y depende fuertemente de la fuente; por lo que cualquier cifra de mérito será para *'ese set de datos'*.

Resultaría utópico un experimento a gran escala que asemeje el *'interceptar'* un sinnúmero de teclados en español, para obtener material crudo y sin sesgo. Igualmente, y en caso de hacerse un experimento puntual, tal vez no se obtenga tampoco el número de datos que represente el estado, distribución y estadística de los errores de la población total que intenta escribir texto castellano/español.

De hecho evidencias de los cursos de español para nativos americanos que se dictan mucho en los EEUU, muestran que el tipo de errores de ortografía cometidos por los hablantes no-nativos hispanos, dista mucho de los errores convencionales que pudimos hallar y tal vez ameritaría un ajuste profundo de los algoritmos para ese uso.

Estrategia de Creación de Conjuntos para Pruebas

Se han realizado varias aproximaciones para determinar un conjunto de pruebas aceptables, siendo el más representativo el de los errores de ortografía más frecuentes. Sin embargo se nota, que se pueden sesgar fácilmente los resultados mediante una elección engañosa, o por lo fortuito de las coincidencias de los errores con las heurísticas de uno y otro método de corrección.

Limitación Natural

Precisamente el tema más importante, partiendo de las hipótesis de esta tesis, es que el sistema a crear está orientado a textos y palabras ingresadas por personas.

Si pensamos en palabras reales y existentes, las palabras resultantes con errores no caerán muy lejos de las palabras que se intentaron escribir originalmente, en términos de distancias de edición, fonética, o cognitiva. Recientemente numerosos trabajos [69] han mencionado y discutido precisamente este tema, brindando pistas de calidad para la elección certera de conjuntos de datos estadísticamente significativos.

Hipótesis de Muestreo

Supongamos que se puede conseguir un "*corpus*" de una enorme cantidad de escritos de toda índole del español. Si este corpus proviene de textos editados profesionalmente como libros, diarios, revistas y publicaciones periodísticas o científicas; no contendrá errores "*brutales*" sino más bien errores sutiles y finos como el mal uso de palabras en contexto además de otros de estilo y concordancia, sumando algunas palabras que fueron mal escritas por no estar incluidas en los diccionarios de las herramientas de edición usadas en ese momento, inclusive a veces las herramientas de edición hacen 'autocorrección' y ponen una palabra similar a la que el escribiente quiso poner. Estos últimos errores son extremadamente difíciles de hallar hasta por el propio autor.

En cambio si el texto obtenido es presumiblemente no editado en forma profesional, sino escrito en la *web 2.0* como redes sociales, mensajes instantáneos y/o e-mails; la abundancia de los errores y su significancia y será completamente diferente.

Finalmente, si el conjunto de los documentos considerado como corpus está balanceado, sus orígenes están suficientemente bien repartidos y su número es grande; podemos suponer que constituirá una muestra significativa del "*estado del arte*" de la escritura humana. Esta misma incluiría errores de dos tipos: sutiles y groseros; no olvidando un componente que son los errores complicados de hallar, esos mismos que tampoco han sido hallados por profesionales de la edición.

10.6 Comparación con el estado del arte

Para poder medir el sistema creado, se debe al menos contrastar con los sistemas que se encuentran en el mercado actual, representando de algún modo el "estado del arte".
A continuación se presentan los sistemas escogidos para tal función.

Se evaluaron varios sistemas de corrección, escogiendo los más importantes referentes del mercado por su difusión y cantidad de usuarios: ***MS-Word 2003*** para productos comerciales de fuente cerrada y por el otro lado, los de fuente abierta y gratuitos.

A continuación se describirá una breve reseña de cada uno junto a la técnica usada.

MS Word 2003 (comercial)

Se considera este editor de texto como uno de los más usados y con una larga historia de versiones a lo largo de casi 20 años; su corrector ortográfico se considera maduro.

Para probar este producto, se construyó un programa que utiliza el MS-Word versión 2003 instalado en la PC de prueba y se accedió directamente a las funciones internas de corrección, usando tecnología exclusiva de Microsoft, llamada **`COM`**, bajo entorno Windows de 64 bits, utilizando ***Interop***, con ***C#*** bajo el entorno **`.NET`**.

Se determinó que la comprobación de ortografía con el ***MS-Word*** es cercana a una palabra por segundo, esto implicó limitar el tamaño del set de pruebas. Durante las pruebas se observó que el Word fallaba con palabras de más de 32 letras, fuera de esto las mediciones se realizaron sin inconvenientes.

Productos OpenSource

Se usó para la medición el ***HunSpell*** y el ***OpenOffice,*** para los de fuente abierta.

Si bien hay una serie de otros productos referenciados anteriormente, como el ***MySpell*** y el ***ASpell*** inclusive uno anterior llamado ***iSpell***; todos ellos comparten tanto la metodología como los diccionarios, pues provienen de un mismo producto, que usa métodos idénticos, permitiendo que diccionarios sean reutilizados una y otra vez.

Los diccionarios compartidos, junto con las metodologías similares hacen que los resultados sean inevitablemente semejantes, salvo sutilezas de velocidad general. Consecuentemente son similares en precisión pero diferentes en performance y en el número de palabras sugeridas.

Se notó también que ambos métodos, y por consecuencia el algoritmo común, poseen fuertes fluctuaciones en sus tiempos de corrección, siendo esto dependiente de los datos, cosa no deseable en lo absoluto para un producto de uso desatendido. Se estima que esto último puede ser debido a la manera en que están implementados los algoritmos.

El algoritmo ***ASpell*** usa metodología de 'fuerza bruta' y *analiza muchas operaciones de edición, sin ponderación ni priorización*, entregando una lista de opciones numerosa. La posición de la predicción correcta frecuentemente no está en la primera posición y el algoritmo tampoco limita el tiempo de ejecución, que resulta poco predecible.

Open Office (gratuito)

El procesador de texto de código *OpenOffice* utiliza internamente una variante del corrector ***Aspell***, que es el más avanzado y difundido corrector ortográfico del ambiente del software libre que se tenga noticia. Se usó la versión: **3.41** de fines del año 2013

Este corrector evolucionó, superando a todos sus antecesores como *MySpell*, e *ISpell*. Se lo considerará como el referente principal del código abierto para las pruebas, siendo esperable que sea el más optimizado por su amplia comunidad de desarrolladores.

Open Office tiene compatibilidad con un formato estándar de diccionarios y hay una enorme cantidad de usuarios que los actualizan periódicamente. Posee una amplia cobertura de idiomas. Su desempeño, sin embargo, esta íntimamente atado a cada diccionario y las diversas particularidades de cada idioma como su grado de inflexión.

Hunspell (comercial/pago)

Esta es una biblioteca nativa **`.NET`**, de la empresa alemana *Maierhofer*[56]. No arrojó diferencias significativas de precisión con el Open Office, solamente es un poco más lenta (20-30%). Esta biblioteca es comercial pero se puede probar gratuitamente.

Hunspell está basada en *Aspell*, siendo compatible con los diccionarios de *Open Office*, por lo cual la cantidad de errores y comportamiento es esperable que sea muy similar.

Este es uno de los pocos productos que posee una considerable difusión y uso comercial, con una amplia cobertura de diccionarios, por la compatibilidad con *Open Office*.

Algoritmo Propuesto "puro"

Se preparó una versión del algoritmo "puro" sin el lematizador ni otras secciones que atacan eficientemente problemas más complejos; para poder ver el comportamiento sin interferencia competitiva de otras partes de corrección como la fonética.

Dado que en el algoritmo es posible ajustar parámetros como el número de correcciones esperadas, se ensayaron varios valores del mismo, no notando diferencias significativas de precisión, pero sí de desempeño en velocidad.

La ventaja del algoritmo propuesto en este trabajo, es la flexibilidad para adaptarse a entornos de poco uso de memoria o procesador, pudiendo ajustar velocidad versus la precisión.

[56] http://www.maierhofer.de/en/open-source/nhunspell-net-spell-checker.aspx

Esto permite sintonizar una prueba para ponerlo similar en cierto parámetro a algún producto de contraste y validar las otras características claramente.

En otras palabras, si ajustamos al algoritmo para tener similar precisión a la del Word, solamente tendríamos la velocidad como cifra de comparación, simplificando todo análisis comparativo.

Algoritmo Propuesto, con Lematizador

Se ensayó una versión completa, que incluye el lematizador (analizador morfológico).

Dado que este algoritmo posee la flexibilidad de poder ajustar parámetros en la cadena de procesos, como el *número de opciones esperadas, incluyendo alófonos,* visto en la **sección 7.1**; primero se realiza un análisis morfológico inicial robusto, intentando restaurar errores diacríticos y si esto falla o el número de palabras obtenidas no alcanza el umbral estipulado, se realiza la restauración ortográfica completa.

La política de estas decisiones en esta cadena es lógica: primero se realiza lo más económico en recursos y que restaura los errores más frecuentes, luego se avanza con métodos más agresivos como el fonético, y finalmente la restauración morfológica. Ni bien se obtienen el número de opciones exigidas, el sistema termina, logrando mejor velocidad.

10.7 Descripción de los Conjuntos de Pruebas

Para poder realizar las pruebas transversalmente, se ha usado un formato compatible de listas de palabras, cada línea encabezada por la palabra mal escrita (con error), y separado por un tabulador la palabra correcta, luego, opcionalmente se puede colocar un tabulador, seguido de un número decimal que indica la frecuencia de aparición de esa forma escrita. Esta frecuencia se utiliza luego para obtener una medición ponderada. El archivo es texto codificado en *Latin-1* (*ISO-8859-1*) o en *UTF-8,* indistintamente.

A continuación se enumera y describe brevemente una lista de sets/conjuntos de prueba.

Set Wiki

Este es un set de prueba de menos de 500 palabras, obtenido de un listado de Wikipedia en español, el cual es referenciado en varios trabajos por lo cual se decidió considerarlo para comparar las cifras de mérito mencionadas en ellos y compararlas. Si bien no es un set muy vasto, contiene palabras que son muy frecuentemente mal escritas. Hubo que eliminar de estas palabras todas aquellas que no son factibles de ser corregidas por este algoritmo, como las palabras partidas. Recordemos que no se propuso hacer un sistema de soldado o re-unión de palabras con espacios mal puestos.

Set 1k

Este es un set de prueba de un poco más de mil palabras, obtenido de un listado extendido de palabras del español frecuentemente mal escritas, obtenidas condensando varios sitios de la web, como los que se han usado en las pruebas del ***ASpell***.

Este conjunto comparte gran número de palabras con el set ***Wiki***, y probablemente hayan sido construidos uno basado en el otro.

Se han agregado numerosas palabras, fonéticamente incorrectas y expresamente mal escritas, "rompiendo" palabras conocidas mediante una o más operaciones de edición. Esto se hizo para analizar el comportamiento del sistema de corrección.

El contenido de este set, es en definitiva un conjunto muy problemático con errores de todo tipo, pero mayormente del tipo cognitivos, los cuales son los más complejos de reparar y por cierto bastante frecuentes, después de los de tildes o diacríticos.

Set 18k

Dentro del marco de este trabajo se elaboró otro set de pruebas más extenso llamado Set 18k, extrayendo palabras mal escritas del listado de palabras únicas, del corpus latinoamericano LIFCACH-2 [38] de textos con 450 millones de palabras obtenidas de diversos orígenes, que fue publicado en el año 2012 en Chile. Este listado de palabras únicas es de libre acceso y contiene el primer millón de palabras más frecuentes y sus frecuencias. Si bien hay otros corpus españoles para usar como el CREA de la RAE, escogimos éste por tener 3 veces más cobertura y estar muy lleno de errores.

Es importante notar que este listado, según los autores, fue construido con un analizador morfológico de la compañía noruega ***Connexor***[57], el cual ha segmentado y etiquetado los términos a partir de los documentos del corpus. Se han observado gran número de errores de segmentación y etiquetado.

Para construirlo, se comenzó por procesar el millón de formas escritas únicas mediante un programa, que forma parte de este trabajo, el cual determina para cada forma, si es viable como palabra, pronunciable en ese idioma y no contiene caracteres especiales.

Finalmente el listado 'resumen' de las palabras obtenidas en esta etapa, fue del orden de 80 mil palabras 'limpias'. Es de destacar que 9 de cada 10 palabras del corpus citado, no eran siquiera palabras, sino meramente formas escritas, mal segmentadas y etiquetadas.

Luego se realizó un análisis más fino de las 80 mil 'supuestas palabras', descartando las palabras correctamente reconocidas por nuestro analizador morfológico [17]. Se obtuvo un listado que supuestamente poseían errores, pues no figuraban en nuestro diccionario; creando así una importante muestra de palabras *"supuestamente erradas"* surgidas de un origen altamente representativo de los errores del lenguaje presentes en textos digitalizados, de enorme diversidad de orígenes y en idioma español.

[57] Connexor Natural Knowledge http://www.connexor.com/nlplib/

Se suprimieron los términos compuestos, incluyendo guiones y apóstrofes, por no ser palabras simples en español y debido a que el algoritmo de corrección propuesto no prevé la corrección de palabras compuestas; también se suprimieron un puñado palabras, que siendo, por ejemplo, modismos o nombres propios, no se hallan en ninguno de los diccionarios, y serían erróneamente consideradas mal escritas.

Se obtuvieron finalmente 18 mil palabras con errores y se procedió a la clasificación automática en palabras con *errores diacríticos*, números, siglas alfanuméricas e irreconocibles, mientras que el resto (*palabras no clasificables*) se revisó en forma manual, eliminando toda palabra presumiblemente inexistente.

Un análisis de los datos obtenidos luego de las clasificaciones, revela que en su gran mayoría (*entre 80 y 90%*), los errores fueron fallas en las marcas diacríticas (*acentos, diéresis, tildes*) coincidiendo con lo observado por diversos autores [46] [73] [75] [81].

Esta sola prueba de similitud estadística nos sugiere que se ha realizado bien el proceso de conformación del conjunto de pruebas y que muy probablemente sea fielmente representativo de los errores en el corpus LIFCACH2 de 450 millones de palabras.

Set SMS

Para aplicar los algoritmos a los textos ingresados en los dispositivos móviles, se necesitaba obtener un conjunto de palabras representativo lo más limpio y directo posible.

Para esto se accedió a un set de mensajes de texto reales de una compañía telefónica de Argentina, cedidos por la empresa ***PandoraBox***[58], para uso estadístico y privado.

Consiste de 182 mil mensajes de texto de hasta 150 caracteres que fueron enviados desde celulares reales a un servicio de consultas comerciales, durante un el año 2013.

Estos mensajes constituyen un excelente muestreo de textos efectivamente escritos por usuarios con dispositivos móviles (*celulares, tablets, smartphones, etc.*) los cuales tienen mayormente mecanismos de ingreso T-9, teclados incómodos a botón y tipo táctil, con algoritmos predictivos.

A pesar de que estos textos puedan contener palabras corregidas por los celulares, se estima que ésta es igualmente una muestra muy representativa del texto que recibiría un sistema inteligente para atención automatizada. Sin dudas este tipo de sistemas requeriría de un algoritmo análogo al presentado para poder procesar y ‘entender’ el texto, sin intervención humana.

La segmentación inicial produjo 1.409.868 formas escritas; luego de un proceso de aglutinación y conteo, se redujo a unas 60 mil formas escritas únicas.

A continuación se realizó un proceso de selección/cribado automático, manteniendo las formas que eran pronunciables o candidatas a ser palabras mal escritas. Preferentemente

[58] ***PandoraBox www.pandorabox.com.ar***

se eliminaron las formas escritas de mínima frecuencia de aparición, que contuvieran número telefónicos, códigos propios y jergas de las empresas de telefonía.

Luego se procedió a aplicar varios algoritmos de restauración y selección muy agresivos, para extraer las palabras candidatas (*corregidas*) en forma semiautomática.

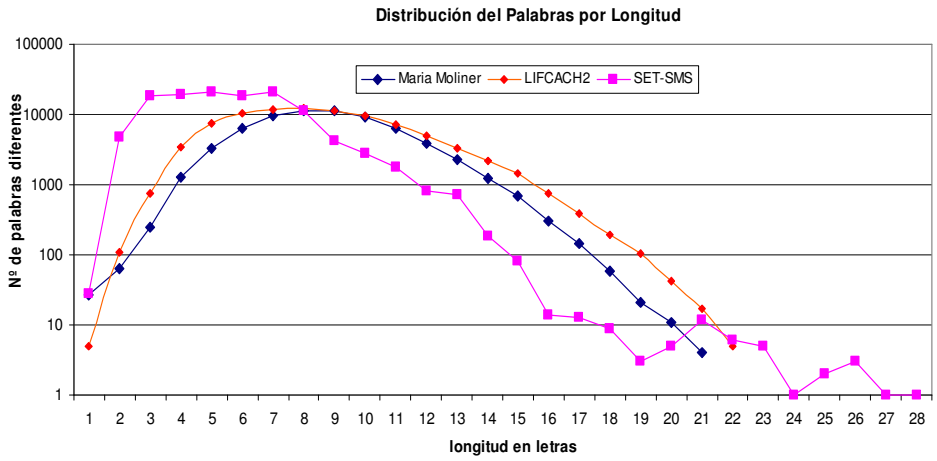
Se editaron manualmente el 80% de las palabras más frecuentes, y se eliminaron palabras imposibles de predecir, incomprensibles o disonantes, nombres propios, extranjeros, marcas, números, etc. El set incluye palabras desde 1 hasta 50 letras, se eliminaron un puñado de mayor longitud para evitar ruido. En este proceso de eliminación, también se descartaron los de más alta frecuencia de aparición y cuya interpretación resultara demasiado ambigua o imposible.

El resultado final fue poco más de 17 mil palabras únicas y 'mal escritas', acompañadas de sus correspondientes correcciones obtenidas en forma semiautomática, incluyendo algunas opciones, consignando la frecuencia de aparición de cada palabra "mal escrita".

Ellas representan 125 mil errores reales (dadas las repeticiones) y corregibles, partiendo de este corpus conteniendo poco más de 1.4 millones de palabras escritas en crudo.

Se podría considerar como lo más parecido a un Gold-Standard de palabras en español ***muy mal escritas*** provenientes de SMS, con su interpretación más probable sin contexto.

Fig 12



Número de palabras en varios corpus, ordenadas por longitud en letras

La distribución de la longitud de palabras obtenidas del corpus, se muestra en la Fig. 11

El corrimiento a la derecha de 3 letras se debe fundamentalmente a que los diccionarios presentan palabras únicas mientras que en el corpus se presenta la cantidad de aparición total de no-palabras y una misma palabra correcta, aparece mal escrita más de 10 veces en formas diferentes, para ciertos casos de palabras muy frecuentes y cortas.

Concluimos que su forma general se asemeja lo suficiente a las de los estándares de diccionarios clásicos presentados anteriormente, para ser considerable estadísticamente.

Analizando detenidamente la relación entre las palabras corregidas por un corrector como el MS-Word, se determinó la siguiente proporción de cada tipo de error detectado.

IgnoreCase	96	0.09%
UnknownChanges	3622	3.4%
ExtraChars	6384	6.1%
RestoreDiacritics	46706	44%
BadChars	48511	46%

Total de errores hallados de cada tipo

IgnoreCase: indica error de colocación de mayúsculas como en un nombre propio,
BadChars es señal de que se ha cambiado una letra por otra,
ExtraChars resulta cuando ha sido necesario agregar una o más letras,
RestoreDiacritics indica que ha sido reparado un error de acentuación escrita,
UnknownChanges indica que ha sido un error no clasificable unívocamente.

Esto es útil para determinar cómo se comportará el corrector ante un conjunto de palabras de estas características, si se conoce de antemano su comportamiento. Incluso es útil conocer la distribución del tipo de errores para poder crear sets o conjuntos de palabras falladas estadísticamente equivalentes, lo cual veremos en la próxima sección.

Necesidad de un Set Artificial

Se observó que la aparición de errores de ortografía en corpus escritos, específicamente los del tipo cognitivos, de distracción, de tipeo e inclusive intencionales fue realmente escasa. Esto se puede deber a que la mayoría de los corpus obtenidos de publicaciones son texto editado, y no contienen las palabras tal cual fueron ‘escritas’ sino que ya poseen, sin duda alguna, una o más revisiones; ya sea por parte del escribiente en el acto mismo de crearlo o del editor en caso de medios gráficos.

Esta asistencia a la escritura, seguramente ha ocurrido con herramientas de corrección ortográfica clásicas como el MS-Word y el Open Office, en sus diferentes versiones, siendo precisamente éstas, por ser las más usadas y comunes, las evaluadas aquí.

Realmente pocos de estos tipos de errores como los cognitivos y de distracción fueron observados en los corpus clásicos, mas no en todas sus diversas formas.

Sólo aparecieron con mayor frecuencia algunos de estos errores en los extractos obtenidos experimentalmente de las redes sociales 2.0 (*twitter, facebook, etc.*) y en especial en el set de mensajería instantánea o SMS.

En virtud de todo esto, y para ensayar extensamente los algoritmos de los productos que pueden representar el estado del arte y contrastarlo con el algoritmo aquí propuesto, es que se ha construido intencionalmente un conjunto de palabras ‘rotas’, tratando de recrear este tipo (cognitivos, de distracción, de tipeo e inclusive intencionales), más frecuentes durante los actos de escritura ‘en crudo’, es decir, sin ‘sugerencias’ ni correcciones automáticas.

A estas palabras 'rotas', se le incluyeron todas las variantes, para probar en profundidad los algoritmos y ver claramente si los demás correctores del mercado contemplan y corrigen estos tipos de errores, dada su escasa o nula frecuencia de aparición en textos publicados, probablemente editados con esos correctores.

Set 10k - Artificial

Los criterios de armado de este conjunto o 'set' artificial de palabras se enuncian a continuación. Se partió de un listado de 110 mil palabras, en su gran mayoría raíces (*lemas*) obtenido de la síntesis de varios diccionarios del español [59].

Dado que se tenía la clasificación de cada palabra, se pudieron descartar: *palabras extranjeras, siglas, abreviaturas y nombres propios v.g. apellidos*, puesto que muchas de esas palabras no poseen regla alguna de fonética ni estadística viable de ser aplicada.

Se excluyeron palabras cortas de 1 y 2 letras, por ser demasiado ambiguas, si bien son muy frecuentes son poco representativas de errores, ya que no se puede concebir como un error el cambio de una letra en una palabra de una sola letra, y en el caso de palabras de dos letras, el cambio es demasiado significativo para el universo de palabras existentes de 1, 2 y 3 letras, pudiendo fácilmente resultar en otra palabra viable y su número es, para las estadísticas, muy poco significativo (menor a 10^2 en 10^5 palabras).

De las restantes, se seleccionaron palabras al azar y a cada una de ellas se le aplicó al azar, con probabilidad del 50% una de las casi 5 mil flexiones/derivaciones de entre 655 prefijos y 4271 sufijos; el resultado se acumuló en un la lista de palabras candidatas para 'romper'.

Posteriormente se aplicaron proporcionalmente cada uno de los siguientes algoritmos de 'rotura' intencional, con proporciones estadísticas establecidas.

Inserción de una letra (insert)
Deleción de una letra (delete)
Inversión de dos letras (swap)
Cambios diacríticos (acentos/eñes/diéresis)
Reemplazos tipo 'qwerty' (replace)

El resultado fue un conjunto de exactamente 10000 palabras con distribución estadística de errores a voluntad, originado al azar desde un universo de más de 10^9 palabras[60].

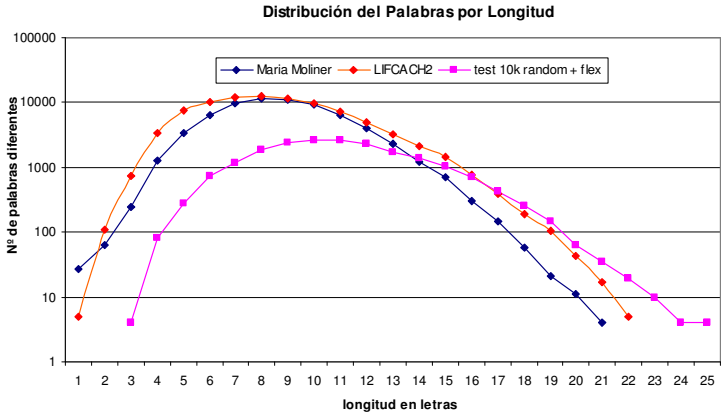
La selección de las proporciones estuvo basada en un promedio aproximado de los hallazgos de errores en corpus, balanceado a favor de más errores accidentales.

[59] El diccionario usado contiene palabras de las siguientes fuentes: Real Academia Española, Larousse, Lexi-K, María Moliner, MS-Encarta 1997, Diccionario Salvat, Diccionario de términos Lunfardos de Clarín, Diccionario de Neologismos de la Academia Argentina de Letras, Diccionarios Médicos: HL-7, CIAP-2, CIE-10, SNOMED en español, CREA, Ancora, Wikipedia en español, WikiDictionary, AGROVOC (taxonomías biológicas), etc.

[60] El total resulta de 100 mil palabras flexionadas, considerando usar solo un 20% de las 5 mil flexiones posibles, sin tener en cuenta la anidación de prefijos y sufijos que resultaría varias órdenes de magnitud mayores.

Para analizar la calidad estadística de la muestra usada, se presenta la distribución de longitudes de palabra del set de prueba empleado, contrastada con la distribución de las mismas en un diccionario clásico de María Moliner y en el corpus *LIFCACH2*.

Fig 13



Número de palabras en varios corpus, ordenadas por longitud en letras

NB: La longitud promedio de las palabras fue de 11 letras (pico de la curva), resultó superior a la de textos convencionales del castellano, que es cercana a 8 letras. Esto implica que el esfuerzo de búsqueda de corrección en el espacio de exploración es importante, dado que el número promedio ponderado de opciones por palabra a corregir[61], es $8.24 * 10^{34}$

El corrimiento hacia la derecha de la curva indica que se ha usado 50% de flexiones y debido a la naturaleza aglutinante y aditiva de estas, la longitud promedio resulta mayor en aprox. 3 letras, como lo indica el comentario anterior.

Este set artificial, dado a que no se origina de un corpus, no posee estadística de errores, por lo que las mediciones serán absolutas, no ponderadas. La única ponderación se realizó balanceando los porcentajes de los tipos de errores creados, tratando de reflejar la estadística promedio de numerosos corpus de castellano.

---

[61] *Este número se halla sumando el producto del número de palabras de cada longitud por el tamaño del espacio, dividido por el número total de palabras*

11 Procedimiento de Medición

Cada set de pruebas constó de un archivo de texto plano, descripto en la **sección 10.7**

Las pruebas con los diferentes productos se realizaron en modo automático en el mismo equipo y se midieron los tiempos de los procesos para determinar su velocidad.

Cada producto seleccionado recibió la palabra ‘mal escrita’, entregando luego del proceso de corrección interno, una lista ordenada de las opciones corregidas; se determinó si la palabra correcta estaba entre ellas, y en caso afirmativo: en que posición.

Se registraron simultáneamente el tiempo de ejecución individual y promedio. Luego se analizó la distribución de tiempos de corrección promedio por longitud de palabra y tipo de error. También se registraron los máximos y mínimos. El resultado se almacenó en una base de datos, para realizar las estadísticas y conteos. Tanto los listados de palabras usadas como así los listados obtenidos a partir de las corridas de testeo, están guardados y disponibles.

Medición Directa y Ponderada

Conforme se presentó en la **sección 10.7**, si los errores ortográficos se obtienen a partir un corpus, se poseerá la frecuencia de aparición de cada forma escrita. En este caso se puede calcular las cifras de mérito de dos manera diferentes. La más simple es número de palabras únicas, o frecuencia absoluta. Por el otro lado y dado que hay errores que se repiten habitualmente, se pueden contabilizar todas y cada una de las apariciones de un error, no importando si ya apareció antes. Esta manera predecirá el comportamiento en el mundo real, puesto que los 'corpus' reflejan justamente eso, la realidad. resumiendo habrá dos mediciones: la absoluta (palabras únicas) y la ponderada (palabras con sus frecuencias de aparición en corpus).

Sería de esperar que la medición absoluta es la importante, puesto que mide cuántas palabras 'diferentes' puede corregir de un set, sin embargo esto no es una representación fiel del comportamiento ante un texto común; mientras que la medición ponderada indicará el porcentaje de corrección o las bondades, basado en la estadística de las palabras erradas, ajustándose más al comportamiento con texto escrito por la gente.

La importancia de la medición ponderada se puede inferir del siguiente caso ficticio: imaginemos que hay solamente dos clases de errores posibles, los de tipo A y los de tipo B, y el sistema corrige el 90% de los tipo A y solo el 30% de los errores tipo B. Si no conocemos de antemano la estadística de la aparición de errores A y B, podríamos decir que el corrector se comportará corrigiendo el promedio de ambas, o sea el 60% de los errores. Sin embargo si resulta que la estadística de un conjunto de pruebas es que el 80% de los errores son de tipo A y solo el 20% son de tipo B, la bondad será mucho mejor: $0.80 \times 0.90 + 0.20 \times 0.30 = 78\%$. Mientras que si la estadística fuese inversa daría $0.20 \times 0.90 + 0.80 \times 0.30 = 42\%$ ¿Cuál sería la bondad correcta para considerar?
La respuesta ' teórica' sería 60%, sin embargo si se trata de usar la caracterización de comportamiento para obtener una predicción, la medición de bondad ponderada se

ajustará más al resultado obtenido en la realidad, siempre que los datos respeten esa estadística y éste resultará mucho más útil, por cierto.

Igual criterio ha sido utilizado en la confección de los sets de pruebas, tratando de reflejar las estadísticas de los tipos de errores, conforme corpus reales.

En consecuencia en toda medición, se indicará si se realizó en forma ponderada o absoluta; siempre y cuando el set haya partido de un corpus y se tenga su estadística.

11.1 Cifras de Mérito

Hay que tener bien claro que se está evaluando un corrector de errores, y para definir una cifra de mérito, el resultado deseado es no tener errores, entonces corresponde contabilizar los 'errores del corrector', o sea las palabras no corregidas.

Veamos un ejemplo, si un sistema A corrige $P_A = 90\%$ de los errores y otro B corregirá $P_B = 80\%$, lo importante aquí es cuántos errores dejan sin poder corregir.

Es decir el corrector A *dejará pasar* $(1 - P_A) = 10\%$; mientras si usáramos que el corrector B *dejará pasar* $(1 - P_B) = 20\%$, justamente el doble del otro.

Tiene sentido entonces plantear como mérito de A respecto B, llamándolo Q_{AB} como inversamente proporcional a los errores que *deja pasar* A, y directamente proporcional a los que *deja pasar* B, el otro:

$$Q_{AB} = \frac{(1 - P_B)}{(1 - P_A)}$$

En este caso A es *el doble de bueno* que B, si corrige el doble de palabras de las que corrige B, lo cual confirma la percepción:

$$Q_{AB} = \frac{(1 - 0.80)}{(1 - 0.90)} \rightarrow 2$$

Otra cifra que se consideró importante es cuantas veces la *"primera palabra* propuesta es la *correcta"*, respecto al total de palabras en cuya lista de opciones propuestas está la correcta. Esto mide realmente cuán bueno es el algoritmo de predicción global, ya que se pueden conseguir hallar cualquier palabra a fuerza bruta de permutaciones, en cambio lograr como primera opción la palabra correcta es otra cosa muchísimo más compleja y pone en juego la calidad y atino de los algoritmos. La fórmula responde al mismo criterio de la anterior, usando las precisiones relativas.

$$Q_{AB1} = \frac{(1 - P_{B1})}{(1 - P_{A1})} \qquad P_{X1} = \frac{(\text{Nº 1ª opción correcta})}{(\text{Nº total corregidas})}$$

Donde el subíndice: P_X indicará tanto el corrector A como el B.

11.2 Utilitario para realizar las Pruebas

En el marco de este trabajo se construyó un programa de testeo y estadísticas, para poder caracterizar los algoritmos como una 'caja negra', aplicando las mismas condiciones a todos. Se escribió en C# para correr bajo Windows con entorno **.NET**
A continuación se analizan y discuten los resultados de las pruebas, explicando algunos detalles del software utilitario construido para las diversas mediciones.

11.3 Pruebas de Calibración

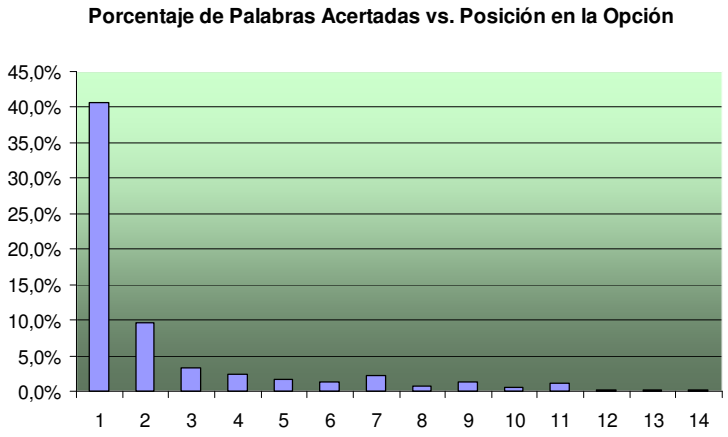
Inicialmente se realizaron dos pruebas de contraste con los sets de palabras conocidos, y los productos citados en trabajos de la literatura, a fin de determinar la calidad de los métodos de medición para con los trabajos referenciados, posteriormente se pasó a un análisis más profundo, considerando que "el instrumento", estaría calibrado.

Se ha contado solo con las listas de ***wikipedia*** en español enriquecidas manualmente, las generadas automáticamente y las extraídas de los trabajos de *Sadowsky-Gamboa* [38]. Resultó imposible conseguir los listados de errores en español citados en el trabajo de *Bustamante-Díaz* [75] para contrastarlo. Este es, en consecuencia, el **set Wiki** que se usará en las siguientes secciones para cada producto de contraste.

Word con set Wiki

Se ensayó el listado de palabras frecuentemente mal escritas originadas en el listado de Wikipedia, tal cual se cita en la literatura. Se presentan un listado y un gráfico simple en la Fig. 14 con la efectividad porcentual del corrector para la primera sugerencia, luego para la segunda y así sucesivamente, los resultados condicen con la literatura.

Fig.14



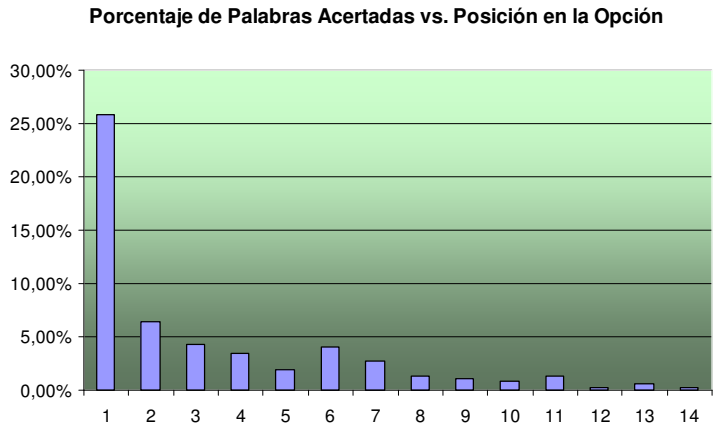
Se observa claramente de la gráfica que el número de sugerencias correctas en primer lugar, si bien es máximo, es escaso; poseyendo una efectividad de apenas 40% para la primera sugerencia, llevando al 66% si se consideran todas las sugerencias (hasta 18).

El resultado obtenido se condice aproximadamente al descripto por *Atserias & all.* [55] a pesar de que el set de pruebas de palabras es diferente.

Open Office con set Wiki

En forma análoga al procedimiento anterior tenemos en la Fig.15, el resultado de la prueba con el Hunspell, que es casi idéntica a la obtenida con el Open Office 3.41

Fig.15



Se observa que este sistema de corrección y sugerencias es muy pobre por su baja calidad y cobertura, llegando a corregir poco más del 25% en el 1º intento y escalando apenas al 54%, cuando se tienen en cuenta todas las sugerencias (*hasta 15*).

11.4 Verificación de la Calibración

Ambas pruebas condicen notablemente con la literatura, las pequeñas diferencias pueden deberse a las versiones de los sets de prueba y las versiones de los productos empleados, que no están explícitamente definidos en los trabajos publicados y por ende no son accesibles. En virtud a esto se consideró válida la calibración.

Los tiempos obtenidos con la herramienta de medición en este trabajo desarrolladas son confiables, dado que están derivados de un reloj interno del sistema operativo que se obtiene a partir de un oscilador de cuarzo, que como se sabe, posee alta precisión y excelente estabilidad intrínseca.

Cabe señalar que no hay mediciones de performance para contrastar, en términos de velocidad, en ninguna de los trabajos de la literatura consultada; curiosamente se omiten éstos y muchos otros datos en todos los productos tanto comerciales como libres a los que se tuvieron acceso.

11.5 Mediciones

Consideraciones Generales

Se presentan el conjunto de las mediciones, consolidadas en tablas numéricas, acompañando algunos gráficos a fin de poder interpretar visualmente el comportamiento, tanto del algoritmo como de los productos contendientes del estado del arte. En cada caso se indicará en color destacado, el ganador en cada categoría.

Los elementos que se deben observar bien son: el porcentaje de palabras correctas en la primera opción propuesta, el porcentaje total de correcciones obtenidas y la velocidad.

A fin de medir los tiempos de cada proceso de corrección se utilizó el reloj interno del sistema, con precisión de 100 nanosegundos y estabilidad de cuarzo, no resultando significativa la indeterminación debida al sistema multitarea y sus interrupciones, dado que su efecto es promediado al realizar los cálculos finales.

El utilitario de medición especialmente desarrollado para esta evaluación, mencionado en la **sección 11.6**, genera archivos de salida testigo con los valores; los cuales están disponibles para contralor de este trabajo.

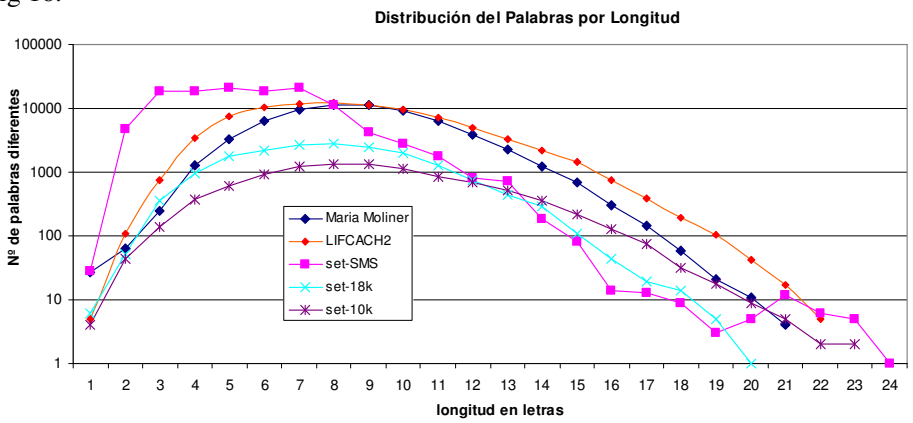
Nombres de Productos y Algoritmos

Se presentan los nombres abreviados usados a lo largo de los siguientes gráficos y tablas a fin de simplificar su lectura.
El parámetro **guess**, usado en las columnas, indica el número de opciones mínimas solicitadas, se ensayaron 1 y 2, dado que un valor mayor no aportaba más correcciones ni cambios significativos pero en cambio ralentizaba el proceso notablemente.

- **Lexer-1** Analizador Morfológico Completo y Optimizado, **guess = 1**
- **Spellah-2** Algoritmo “puro”, **guess = 2** (+greedy)
- **Spellah-1** Algoritmo “puro”, **guess = 1** (lazy)
- **AOO 3.41** Apache Open Office 3.41 AOOm1(Build9593) – Rev.1372282
- **HunSpell** Librería Comercial de *Maierhofer* corriendo en W7 x64 bits.
- **Word 2003** Micrsoft Word 2003, ejecutándose en W7 x32 bits.

Estadísticas

Se presenta a efectos de visualizar la distribución comparativa entre conjuntos.
Fig 16.



Resultados

Se presentarán tablas de mediciones correspondientes a cada set de datos, dado que sintetizarlos en una misma tabla tornaría ésta difícil de interpretar. Se ensayaron los siguientes conjuntos de datos, en el siguiente orden:

Set SMS (Proveniente de 183 mil mensajes de texto reales, de una telefónica Argentina)
Set 18k (Basado en LIFCACH2, origen de 450 millones de palabras Latinoamericanas),
Set 10k (Artificial, creado con proporciones de tipos de errores similares a las reales)

Para todas las tablas de datos, las columnas corresponden a los nombres abreviados de los diferentes algoritmos y productos de contraste, explicados en el listado anterior; mientras que las filas indican los valores obtenidos y algunos cálculos pertinentes.
El principal es **C_T** que es la fracción de palabras que logra restaurar, mientras que **C_1** representa la fracción corregida como primera opción. Las expresiones son:

$$C_T = \frac{N°\,corregidas\_totales}{N°\,total} \qquad C_1 = \frac{N°\,corregidas\_1°\_opción}{N°\,total}$$

La sigla **U_T** (***U**ncorrected **T**otal*) indica el porcentaje de palabras incorrectas que no logran ser corregidas, cifra asimilable al 'ruido' remanente no corregido. Luego, normalizando contra el mejor que será: **U_{TMin}**, obtenemos: **U_N** (***U**ncorrected **N**ormalized*) coloquialmente sería 'cuánto fracción de ruido deja pasar, respecto al mejor' y a valores mayores es peor.

$$U_T = 1 - C_T \qquad U_{TMin} = 1 - C_{TMin} \qquad U_N = \frac{U_T}{U_{TMin}}$$

Análogamente podemos hallar para la fracción de 1° opción correcta $U_{1N} = \frac{U_1}{U_{1Min}}$

La cifra de mérito **S_N** (***S**peed*) indica la fracción de velocidad normalizada respecto al de mayor velocidad; en otras palabras, 'cuántas veces más lento es, respecto al más rápido', por lo que valores mayores indican menor performance, la fórmula es:

$$S_N = \frac{V_{Max}}{V_T}$$

Finalmente **F_T** representa una cifra de mérito total, adimensional y normalizada, usando una media geométrica de las bondades respecto al primer hallazgo y al total, siendo:

$$F_T = S_N \cdot \sqrt{U_N \cdot U_{1N}}$$

Recitaría 'cuántas veces más lento y ruidoso es (en promedio), que el mejor'

Set SMS

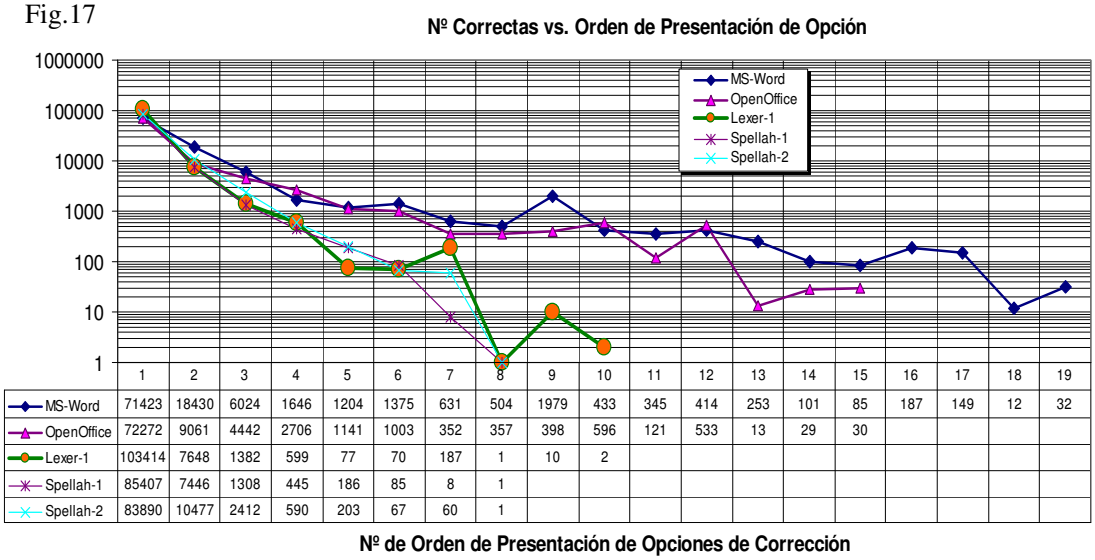
Los resultados obtenidos para el **Set SMS** fueron los siguientes:

		Lexer-1	Spellah-2	Spellah-1	AOO 3.41	HunSpell	Word 2003
C_T	%	90 (85)	78 (75)	76 (72)	74 (64)	75 (65)	84 (71)
C_1	%	82 (51)	67 (56)	68 (61)	58 (42)	58 (43)	57 (47)
V_T	words/s	165.0	9.5	24.4	27.7	19.0	2.7
S_N	(ratio)	1.0	17.4	6.8	6.0	8.7	60.7
U_T	%	9.5%	22%	24%	26%	25%	16%
U_N	(ratio)	1,0	2.3	2.5	2.7	2.6	1.7
F_{TU}	(ratio)	(2.1)	(47.7)	(17.3)	(21.1)	(29.6)	(147.7)
F_T	(ratio)	1,0	36.4	14.5	15.3	21.8	123.4

Nótese los valores entre paréntesis, indican las fracciones absolutas contando palabras únicas, mientras que la cifra primera, indica la fracción ponderada al corpus, contabilizando por cada palabra errada el número de veces que ocurre realmente, el corpus tiene 17185 palabras diferentes, representando un total de 125321 apariciones en el corpus de SMS, de 1.4 Millones de palabras, es decir hay 8.8% de palabras mal escritas, de las cuales hay un promedio de repetición de 7.2 veces cada una. En realidad unas pocas se repiten muchísimas veces, siguiendo aprox. la conocida ley de Zipf [62]. La ponderación aproxima las cifras de mérito al comportamiento de un producto fuera del laboratorio, pues considera estadísticas reales. Se observa que todos los correctores mejoran respecto al conteo unitario, significando que corrigen mejor las palabras más frecuentes, y esto es sin dudas, muy deseable. Se presenta para contraste F_{TU} indicando el factor de mérito total para palabras únicas, siendo mayor que el ponderado.

Este es el más salvaje de los corpus analizados, pues contiene una elevada proporción de errores de ortografía, casi horrores; y como sabemos, es proveniente de usuarios de celulares de Argentina, mayormente jóvenes, de entre 15 y 35 años. Comparando esta tasa con la de corpus contemporáneos del español como el **Set 18k**, se infiere que hay un problema serio de educación en ese segmento etario, muy a pesar de que el método de entrada de texto en un celular no es óptimo e induce a errores, pero nunca tantos.

Fig.17



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
MS-Word	71423	18430	6024	1646	1204	1375	631	504	1979	433	345	414	253	101	85	187	149	12	32
OpenOffice	72272	9061	4442	2706	1141	1003	352	357	398	596	121	533	13	29	30				
Lexer-1	103414	7648	1382	599	77	70	187	1	10	2									
Spellah-1	85407	7446	1308	445	186	85	8	1											
Spellah-2	83890	10477	2412	590	203	67	60	1											

[62] Ley empírica de Zipf, propuesta por George Kigsley Zipf, de la universidad de Harvard postula que la frecuencia de aparición de una palabra sigue aproximadamente la inversa de su número de aparición elevado una potencia cercana a uno

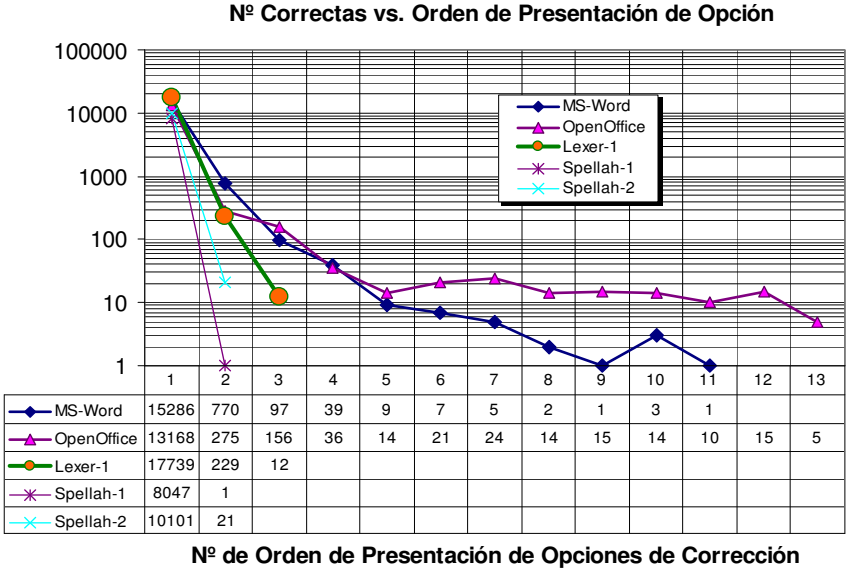
Set 18k

Los resultados obtenidos con el **Set 18k** fueron los siguientes:

		Lexer-1	Spellah-2	Spellah-1	AOO 3.41	HunSpell	Word 2003
C_T	%	**98.5**	55.5	**44.1**	75.5	75.3	83,9
C_1	%	**97.2**	55,3	**44.1**	72.1	71.7	68,0
V_T	words/s	**988.0**	19.3	15.8	42.0	30.4	**3.1**
S_N	(ratio)	1.0	51.2	62.5	23.5	32.5	**315.7**
U_T	%	1,5%	44.5%	55.9%	24.5%	24.7%	**11.1%**
U_N	(ratio)	1.0	30.0	**37.6**	16.5	16.6	7.5
F_{TU}	(ratio)	1.0	1118.7	1712.7	301.3	421.0	**2074.0**

No hay que dejarse engañar por la 'espectacular' velocidad de 988 palabras/segundo del algoritmo compuesto **Lexer-1** dado que este corpus, posee una enorme cantidad de errores diacríticos (*acentos, diéresis y eñes*) y hay un mecanismo propio del lematizador que corrige marcas diacríticas mediante una veloz operación de 'hash', sin necesidad de entrar al algoritmo de corrección especulativo, el detalle se visualiza en la Fig.18. Recordemos que este set fue extraído del corpus de 450 millones de palabras del castellano contemporáneo y latinoamericano, *LIFCACH2*

Fig.18



	1	2	3	4	5	6	7	8	9	10	11	12	13
MS-Word	15286	770	97	39	9	7	5	2	1	3	1		
OpenOffice	13168	275	156	36	14	21	24	14	15	14	10	15	5
Lexer-1	17739	229	12										
Spellah-1	8047	1											
Spellah-2	10101	21											

Una vez más se destaca que lo importante es que la primer opción sea la correcta y éste porcentaje sea alto respecto a todas las corregidas en diversas posiciones de la lista de opciones. Este efecto se ve claramente, puesto que los algoritmos **Spellah-1** y **Spellah-2** presentan muy pocas opciones alternativas, si bien no corrigen todas las palabras mediante su algoritmo. El hecho que **Spellah-1** sea el peor de todos en calidad, se debe a que la mayoría de los errores de este set son diacríticos y el algoritmo de reconstrucción puro, no posee características fonéticas sobresalientes, frente a las reglas de acentuación, puesto que éstas no son estadísticamente relevantes, y son casi arbitrarias; mientras que los demás errores tratables sí las poseen, como se desprende de su buen comportamiento en todos los demás sets de datos, ante omisiones, inversiones y faltantes de letras.

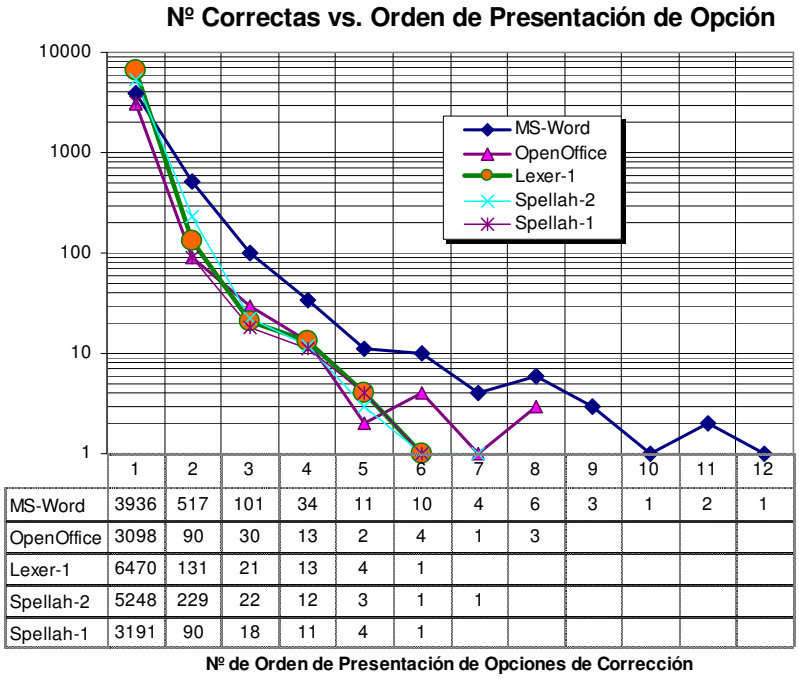
Set 10k

Los resultados obtenidos con el **Set 10k** (artificial) fueron los siguientes:

		Lexer-1	Spellah-2	Spellah-1	AOO 3.41	HunSpell	Word 2003
C_T	%	66.4	55.2	33.2	32.4	32.6	46.3
C_1	%	64.7	52.5	31.9	31.0	31.1	39.4
V_T	words/s	83.0	13.1	13.3	15.0	10.9	3.2
S_N	(ratio)	1.0	6.3	6.2	5.5	7.6	25.9
U_T	%	33.6	44.8	66.8	67.6	67.4	53.7
U_N	(ratio)	1.0	1.3	2.0	2.0	2.0	1.6
F_{TU}	(ratio)	1.0	7.3	8.8	7.9	10.8	32.7

En el comportamiento de las listas de opciones, se presentan AOO3.41 y HunSpell colapsadas como OpenOffice, ya que sus valores casi no difieren.

Fig.19



	1	2	3	4	5	6	7	8	9	10	11	12
MS-Word	3936	517	101	34	11	10	4	6	3	1	2	1
OpenOffice	3098	90	30	13	2	4	1	3				
Lexer-1	6470	131	21	13	4	1						
Spellah-2	5248	229	22	12	3	1	1					
Spellah-1	3191	90	18	11	4	1						

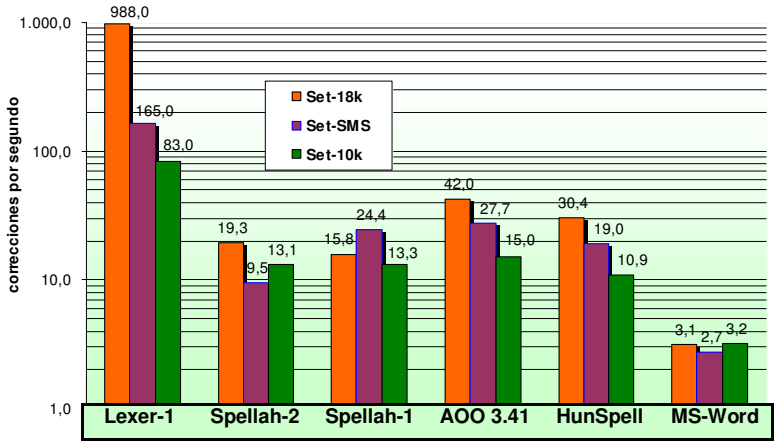
Se observa con este set de datos, que **Lexer-1** domina consistentemente, inclusive en sus variantes 'puras' (**Spellah-1 y 2**) este set de datos conforma un escenario muy extremo, dado que las conjugaciones utilizadas en el 50% del mismo son poco frecuentes, razón por la cual el algoritmo se destacó por la cobertura, mientras que sus contendientes fallaron, sin embargo las cifras de estos últimos son cercanas a los trabajos publicados respecto tanto al OpenOffice y al MS-Word en el trabajo de *Atserias* [55]

Con este set de datos, también se observa, como en todos los anteriores, una clara diferencia a favor de las tres variantes del algoritmo presentado vs. el estado del arte.

11.6 Resumen de las Mediciones

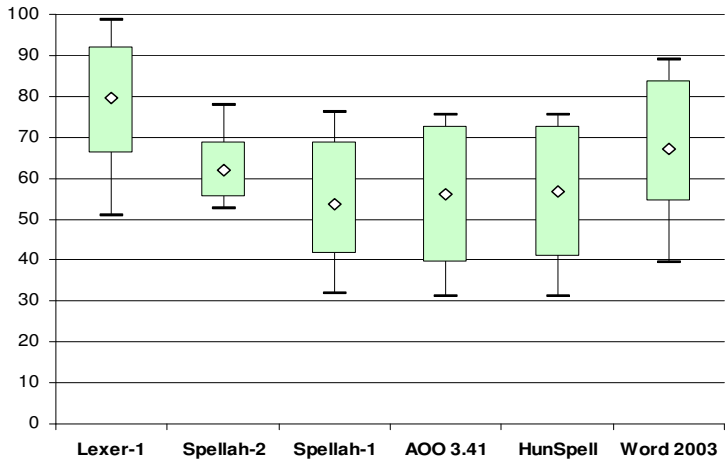
Lo importante de estas mediciones se resume en dos factores: la ganancia de velocidad simultáneamente con la precisión y el porcentaje de aciertos en la primera opción.

Se presenta en la Fig. 20 el diagrama de velocidad de proceso de los tres sets de datos y los seis productos.
Fig. 20



La precisión se unificó para los tres sets de datos, indistintamente, usando un esquema de caja y bigotes, en la Fig.21 indicando los límites y el promedio en forma elegante.

Fig. 21



La lectura minuciosa de las tablas indica que **Lexer-1** representa una '*mejora*' sustanciosa sobre el algoritmo **Spellah** 'básico', superando ampliamente a sus competidores como el ***OpenOffice*** por 8-300 veces, ***Hunspell*** en 10-420 veces y ***Office*** en 32-1600 veces.
Si llevásemos el factor 'veces' a porcentaje de mejora, la lectura sería mucho más extraordinaria aún, pero no se quiere mostrar 'espectacularidad' sino consistencia.

Lo destacable es que las tres variantes del algoritmo, logran ventajas consistentes en la cifra de mérito conjunta, y en ninguna de las cifras parciales deja de tener una ventaja superior a un orden de magnitud, respecto a las que representan el estado del arte.

Estas cifras muestran un factor de calidad extraordinario, sin embargo los resultados son muy sensibles al tipo de muestra, y sin dudas se podrían haber confeccionado en forma engañosa. Pero con el fin de evitar susceptibilidades, las mismas fueron tomadas en forma rigurosamente aleatoria, como se explicó en detalle a lo largo de las secciones. En virtud a esto, podemos asegurar que la medición tiene bajo sesgo, aunque tal vez un poco pueda deberse a que las palabras factibles de ser corregidas, se basaron en nuestro diccionario, el cual tiene diferente cobertura que los de *Open Office* y *MS-Word*.

11.7 Comportamiento Diferencial

Cuando medimos cosas que actuarán sobre un sinnúmero de elementos y condiciones que no conocemos de antemano; como en este caso las palabras; las estadísticas nos entregan mediciones de comportamientos promedio, lo cual no quita que en el acto de promediar, se sesgue y aplanen comportamientos erráticos o inesperados.

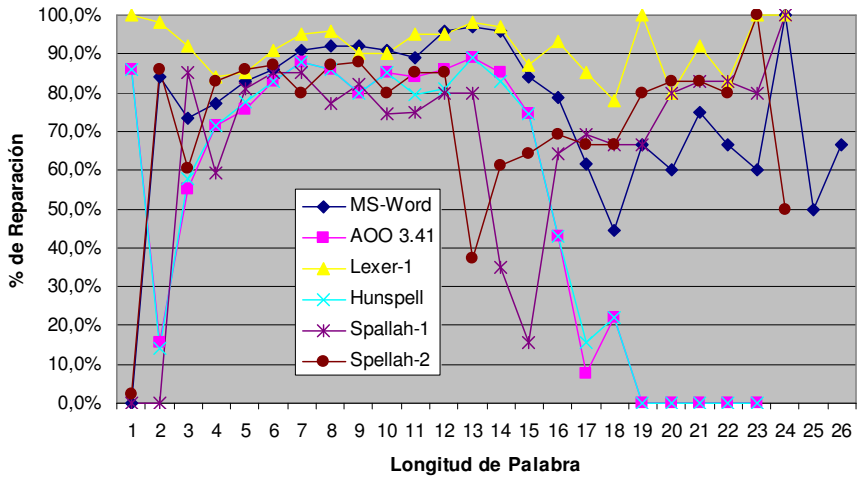
Esto es demasiado cierto en casi toda la ingeniería y de hecho los llamados "bugs", como las fallas aleatorias en soft y hardware, destacando los "easter-eggs" del soft, son muestras demasiado frecuentes de pruebas muchas veces mal diseñadas; algunas veces por ser imposibles de realizar en su totalidad como en este caso de las palabras. Tal es el mal famoso error llamado FDIV-error del primer "***Pentium***" de ***intel***, el cual en 1994 pasó exitosamente una batería de pruebas rigurosas; pero dado que era imposible de probar todas las cuentas posibles de punto flotante, en un tiempo finito, este hecho no se debió a un error indetectable, sino precisamente a un error en la metodología de pruebas.

En relación a esto, una cosa preocupante, es que un algoritmo cualquiera se comporte erráticamente ante diversas situaciones impredecibles, como pueden ser aquí la longitud de las palabras y dado que hemos observado en los productos comerciales una tendencia a demorarse excesivamente corrigiendo determinadas palabras, es que se encaró un estudio limitado del comportamiento que de paso arrojó luces sobre el funcionamiento interno de los algoritmos contrincantes y marcó diferencias interesantes, posicionando de paso muy bien al algoritmo presentado en el marco de este trabajo.

Para esto se estudió el comportamiento de corrección sobre el **Set SMS**, por ser el más complejo y salvaje, por las características de los errores contenidos. El gráfico presentado, muestra que tanto el *OpenOffice* como el *Hunspell* coinciden en forma estricta, avalando aún más la aseveración que hacen sobre que son el mismo algoritmo.

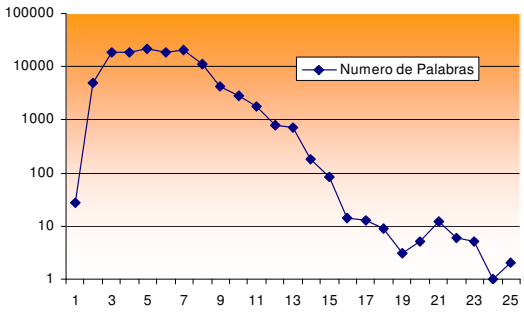
El gráfico siguiente, Fig. 22 muestra un análisis del porcentaje de correcciones exitosas totales, en función de la longitud de la palabra a corregir sobre el **Set SMS**.

Fig. 22



Nótese que los sectores fuera de 3-17 letras no tienen significancia estadística, por no poseer un número suficiente de palabras, dada su siguiente distribución, como se ve claramente en la Fig.23, y se muestra en escala logarítmica, dado su rango dinámico.

Fig. 23



Número de palabras por longitud

Algo interesante resultó al analizar los tiempos de proceso para cada longitud de palabra, que no se muestran aquí; dado que sería difícil de representar.

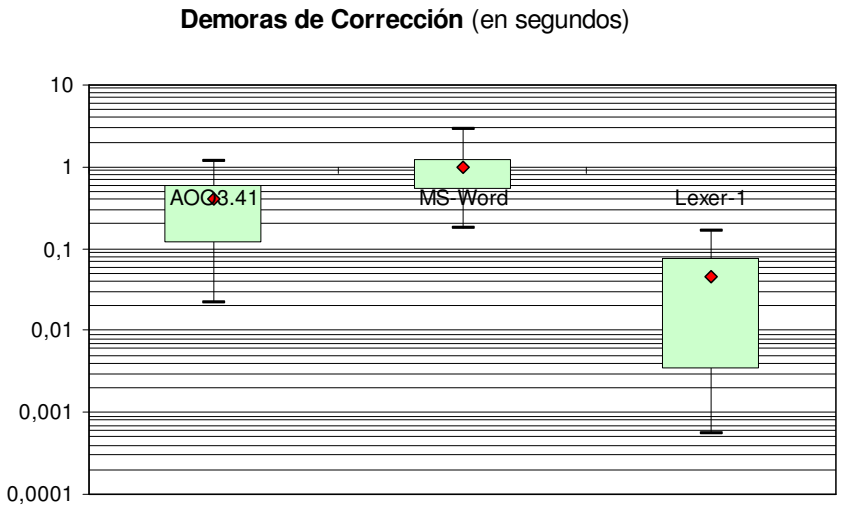
Se notó que casi todos los demás algoritmos tenían tiempos muy dispares, con arrebatos de corrección de varios segundos tanto para el *MS-Word* como para ambas las versiones libres, basadas en Aspell como *Open Office y Hunspell*. Esto se puede deber a que cada producto hace el mejor esfuerzo y como el número de operaciones es combinatorio y tal vez no limitado internamente, da como resultado este comportamiento impredecible.

En cambio el algoritmo ***SpellAH*** se mantuvo dentro de parámetros perfectamente acotados, no llegando en ningún caso a exceder 3 veces el tiempo promedio.

Sin lugar a dudas, este comportamiento predecible se debe al cuidadoso diseño de ***SpellAH***, limitando algorítmicamente el número de intentos de restauración, sin por eso influenciar prácticamente en nada su calidad ni cobertura.

Se presenta finalmente un gráfico de cajas en Fig. 24 con rangos de demora máxima de los tres representantes principales (AOO, Word y Lexer) totalizado sobre los tres sets.

Fig. 24



Rangos de demora de corrección de los tres productos ensayados

12 Discusión del Aporte

Se presenta un algoritmo simple, rápido y eficaz, parametrizado para lograr un ajuste entre precisión, velocidad y uso de recursos; estando su desempeño consistentemente encima de los productos considerados como representantes del presente estado del arte.

El algoritmo presentado al que hemos llamado ***spellah,*** se clasificaría como un sistema que usa el concepto de *boosting no pre-entrenado con competencia dinámica de expertos y búsqueda especulativa optimizada*.

El producto final se consolidó como analizador morfológico ***Lexer*** con capacidad de corrector que emplea ***competencia de expertos***, muy usado en aprendizaje automático; logrando mejor precisión en promedio, que el mejor de ellos; obteniendo una substancial ganancia de performance general, corroborada con las mediciones realizadas.

La ganancia de precisión y performance van de la mano en el diseño, aunque no parezca aparente, dado a que si se logra hacer las cosas muy rápido, sobra algún tiempo para probar otras alternativas sin afectar el desempeño, contribuyendo más con la precisión.

12.1 Ventajas Diferenciales

La mayoría de los correctores que existen en el mercado, son interactivos y presentan una lista de palabras no jerarquizadas, por lo cual resulta difícil determinar "*cuál es la mejor propuesta*". Esto seguramente se debe al diseño pensado en que la corrección sería luego asistida por el usuario. Afortunadamente en todos estos casos, la primer palabra en la lista de todos los correctores analizados, resultó casi siempre la correcta.

Una ventaja de ***spellah***, es la alta tasa de primera opción correcta, además de que provee una lista ponderada de candidatas para ser usadas en desambiguación. Esto permite sin dudas que un algoritmo posterior pueda seleccionar la palabra más adecuada al contexto en juego; en la **sección 11.5** se menciona un ejemplo básico de este uso, que se implementó y con resultado alentador.

Podríamos resumir las características obtenidas del sistema construido, que incluye el corrector de ortografía como pieza vital para su funcionamiento son las especificaciones finales logradas, que se detallan a continuación.

Especificaciones Finales de Lexer-1

- **Velocidad**
  - Corrección Pura: desde 30 hasta 800 palabras por segundo, dependiendo del error.
  - Validación interna del Diccionario Flexivo > 100.000 palabras/segundo
  - Lematización + Corrección: ~ 1000 palabras por segundo, para texto común con una tasa 'normal' de errores (menos del 5%).
- **Tasas de Corrección**
  - Correcta en 1º opción ~ 80 - 98.5%
  - Mérito General > 85 %
  - Diacrítica ~ 99 %
  - Fonética ~ 70 %
  - Simple Letra: 99 %
  - Multiletra: ~ 50 %
  - Separación Número-Palabra
  - Corte de palabras (10kg → 10 kg)
- **Diccionario Flexivo**
  - Ampliable Fácilmente
  - Número de palabras reconocidas > 300 millones
- **Etiquetado Morfológico**
  - EAGLEs 2.0 (ampliado)
  - Probabilidad (en caso de polisemia)
  - Frecuencia estimada en corpus español
  - Verosimilitud (en caso de corrección ortográfica)
- **Reconocimiento de Entidades**
  - Números Enteros (dígitos: 9251)
  - Formato Científico (+10.34 e-12)
  - Hexadecimal (0x12)
  - Cardinales (dos mil ciento treinta)
  - Ordinales (octavo, décimo segundo)
  - Fraccionales (treintavo)
  - Multiplicativos (doble, triple, cuádruple, etc.)
  - 996 Monedas (dólar, euro, dólar canadiense, peso, etc.)
  - 174 Unidades ISO-IANA
  - 1094 Unidades Internacionales (kg, m², Joule, N, etc.)
  - Términos Multipalabra (pájaro carpintero, oso gris, etc.)
  - Formulas Químicas (IUPAC)
    - 121 Elementos químicos con datos generales.
    - 242 Fórmulas, 1932 Identificadores, 893 Nombres, 29 Isómeros
    - Isómeros Estequiométricos (presenta polisemia)
    - Fórmulas en palabras, estimación con 5 reglas (ampliable)
- **Aglutinación de Terminologías**
  - 20604 Locuciones Españolas
  - 1380 Abreviaturas (RAE, ampliadas)
  - Terminología Científica (*Biología, Botánica, Medicina, Química, etc.*)
- **Inferencia de Términos y Nombres Propios**
  - 16833 Nombres de Pila (Personas) Ej: *Dr. Adriano P. Herzógsky*

Extracción Semántica

- Presentación de significado (expresamente presente en el diccionario)
- Estimación en base a sufijos y prefijos (si no hay significado presente)
- Cálculo del número equivalente para el caso de locuciones numéricas

---

12.2 Análisis Comparativo

Las mediciones utilizaron correctores realizados dentro del marco de este trabajo y correctores de sistemas comerciales, y es representativa del estado del mercado consolidado de los últimos 10 años.

Los resultados obtenidos con ***spellah*** solamente son superados en contadas ocurrencias de errores-correcciones por otros productos actuales, y esto sólo ocurre cuando estos productos incluyen gramática y estadísticas del contexto; siendo este tema de corte netamente lingüístico, está claramente fuera de las especificaciones del presente trabajo, y naturalmente podría dar lugar a su extensión o mejora.

Los demás sistemas de corrección considerados, tanto comerciales como gratuitos, no detectan ni corrigen todos los tipos de formas escritas que este sistema realiza, destacándose principalmente: la corrección fonética, el corte de palabras 'pegadas', la interpretación fonética de palabras con números, la detección de elementos químicos y por ende la detección e inferencia de las palabras fuera de diccionario y/o parasintéticas.

Hasta octubre del 2014, no se hallaron productos en el mercado que se acerquen ni que rivalicen con ***spellah*** tanto por las características de reparación como en su robustez, flexibilidad, economía de recursos y velocidad de trabajo.

12.3 Puesta en Valor del Algoritmo

Un algoritmo nuevo, trata justamente de resolver algo que jamás fue hecho, o en caso de haber sido abordado anteriormente, tratará de hacerlo mejor, logrando que sea más veloz, más preciso y/o que use menos recursos, incluida toda combinación virtuosa.

Precisamente ***spellah***, por su diseño e implementación, probó ser eficiente y eficaz, al poseer una alta cifra de mérito para una corrección exitosa en el primer intento, y resultar simultáneamente varias órdenes de magnitud más veloz que el más rápido y preciso hallado en el mercado.

No se midió ni contrastó el uso de recursos pues el de los productos contrastados es muy incierto, por no poder separar que parte del recurso es para corrección y que parte es para otras cosas, ya que son productos muy complejos. Vale la pena comentar que, ***spellah*** posee un uso de memoria sumamente conservativo, pudiendo negociar en bastante rango, mediante el uso de parámetros, su velocidad contra uso de memoria.

Otra cosa que se puede destacar es que ***spellah***, no solo hace una corrección a secas, sino realiza un pormenorizado análisis de las palabras; por lo que si se evitara, seguramente se lograría incrementar la velocidad no menos de un orden de magnitud. Sin embargo el análisis morfológico realizado, resulta sumamente útil para que sea usado en sistemas inteligentes automáticos y autónomos.

Esta propuesta de algoritmo se considera novedosa ya que representa una mejora considerable en la relación de costo/beneficio respecto de los algoritmos del mercado, superando ampliamente las especificaciones previstas para el trabajo.

Todo esto permite concluir que resulta un aporte particularmente valioso, considerando que se aplica al idioma español, que es el segundo en número de hablantes de occidente.

12.4 Aplicación y Uso

La principal ventaja que se estima es su performance, con bajo uso de recursos que posibilita la inclusión de corrección en procesos en donde antes era prohibitivo por su costo y uso de recursos. El lograr mejoras de varios órdenes de magnitud sin dudas permite extender los alcances de aplicación a cosas tal vez no pensadas antes.

El algoritmo creado permitiría su utilización sin muchas modificaciones tanto en servicios remotos del tipo ***SaaS*** (*Software as a Service*) como en sistemas embebidos, en especial aquellos con escasa memoria y limitada velocidad de proceso, prometiendo a pesar de eso, una ajustable y razonable tasa de costo/beneficio.

Ya en ha sido incluido en una plataforma (*framework*) de procesamiento de lenguaje natural robusto [19] usado para sistemas conversacionales en lenguaje natural con el objeto de construir agentes virtuales basados en diálogo hombre-máquina [20].

Conjuntamente con el sistema descripto en [20], se realizó un experimento, usándolo como eslabón de entrada de un producto más avanzado. Se han logrado así correcciones de ortografía de índole contextual/gramatical, utilizando relaciones estadísticas de grupos de palabras y aprovechando las opciones provistas por el ***spellah***.

La corrección mencionada, se consiguió gracias a un proceso posterior de etiquetado morfológico de la función gramatical *POS (Part of Speech*) más probable, basado en el conocido algoritmo de *Viterbi*, aplicado a un modelo de cadenas ocultas de *Markov,* previamente entrenado a partir de un corpus anotado del español, breve pero de muy buena calidad, llamado CastL3B. Algo notable de este trabajo [20], es que se han podido corregir palabras que están bien escritas morfológicamente, pero "suenan" parecidas (*alófonos aproximados*) a las palabras que gramaticalmente son correctas y el sistema las corrige por contexto.

En ambos casos [20] [19] el sistema posibilitó el reconocimiento de textos en bastante mal estado, ortográficamente hablando, como lo son los textos provenientes de mensajería instantánea (chat). Sin dudas aportó flexibilidad a la hora de balancear costo computacional vs. calidad de reconocimiento en tiempo real, logrando analizar y corregir razonablemente bien hasta 50 frases cortas por segundo, por núcleo de CPU.

12.5 Alcances del Algoritmo

Existen aún numerosos problemas de palabras que un humano corrige fácilmente y que ***spellah*** aún no pudo resolver, muy a pesar de la cantidad de políticas de corrección aplicadas. Se enumeran y comentan a continuación las más relevantes:

- Errores fonéticos combinados con diacríticos, en especial en las conjugaciones. Esto se debe a que el algoritmo ortográfico no usa fonética y el fonético descansa sobre la búsqueda aproximada en un TST. Éste fue construido con fonemas derivados de las palabras raíz y no de todas las formas flexionadas posibles. Sobrellevar esta limitación se puede, pero conlleva un espacio de memoria mayor, conjugando todas las formas y acarreando un mayor tiempo de inicialización o tenerlo preprocesado y almacenado. Otra posibilidad es agregar en el diccionario un listado de términos frecuentemente usados en la conjugación, aumentando un poco de uso de memoria pero incrementando el diccionario sólo con las palabras agregadas. Esto mejora sólo las estadísticas.

- Separación de palabras entre las que se omitió el espacio o en su defecto se reemplazó por puntos o guión bajo. Lamentablemente esta situación posee un alto costo computacional. Para evitar este costo, se limita el número máximo de partes en que se seccionará esa forma escrita, pero no se resuelve la separación entre palabras sino en algunos casos, el algoritmo ya lo contempla sutilmente.

- En contados casos de palabras cuyos errores de sustitución de letras no producían 'pares' de letras atípicas, (v.g. amable donde bl es atípico) el sistema presentó primero palabras alternativas que no fueron las apropiadas. Esto se puede corregir tomando en cuenta la frecuencia de aparición en corpus de las palabras con pares atípicos, para generar una selección a favor de las palabras (*enteras*) más frecuentes, en lugar de las 'secciones de n-gramas' más frecuentes, pero el costo de memoria y complejidad asociado a la selección de palabras enteras no se justifica, si se desean mantener las condiciones de recursos computacionales bajos. Esto es económico en términos computacionales, pero costoso en la recolección y limpieza de los datos estadísticos del lenguaje.

- La búsqueda fonética falla en casos de mucha distancia fonética; hay un parámetro interno de radio de búsqueda por cada paso colocado para evitar que ***spellah*** se 'desboque' y explore demasiadas alternativas. Afortunadamente para numerosos casos en los cuales los demás correctores fallan, ***spellah*** sale airoso, mediante el uso de ese parámetro, colocado en una forma conservadora de memoria y tiempo.

- Se notaron en el corpus de SMS una alta cantidad de siglas, acrónimos y palabras en inglés. Esto es el resultado de un sesgo relacionado con la adopción de palabras de marketing como 'smspack' 'bonustrack' junto con los modelos de los teléfonos y sus siglas. Este problema solo se resuelve agregando a los diccionarios del sistema estas palabras y siglas, que no funcionarán con las funciones fonéticas por pertenecer a idiomas diferentes al español.

Comparándolo con el estado del arte actual, le queda suficiente rango para sacrificar performance y aumentar el uso de memoria para obtener muy altas cifras de mérito y crear un producto comercial altamente competitivo.

12.6 Futuros Desarrollos y Mejoras Posibles

Para presentar las aplicaciones y mejoras, se detalla a continuación un listado de opciones, con títulos que reflejan claramente la categoría o rango de aplicación de un posible desarrollo futuro de ***spellah***.

Mejora de Velocidad

Una mejora de velocidad de ***spellah*** es esperable mediante el uso de un sistema de caché de memoria, a pesar de que este tipo de mejora solo es notable cuando el tamaño del caché está en relación con la estadística de errores y palabras de los documentos a tratar, actualmente el tamaño del caché es un parámetro modificable.

Sin embargo, nótese que los sistemas de memoria tipo caché tienen una fuerte influencia negativa en las primeras operaciones, a la vez que en el caso particular de este tipo de datos (*palabras*) no tiene mayor sentido almacenar palabras erradas, pues su cantidad es abrumadora y no se repiten estadísticamente casi nunca, salvo un puñado de errores muy frecuentes. Dado que la frecuencia de aparición de palabras erradas es muy baja, el uso de este recurso sería muy desperdiciador y hasta haría imposible su mantenimiento.

Solamente tiene sentido almacenar palabras muy frecuentes del lenguaje en un esquema de caché circular tipo ***LRU*** (*Least-Recently-Used*) el cual ya ha sido implementado exitosamente en el Analizador Morfológico y funciona bien, llevando la performance general a cerca de 100 veces el valor de velocidad del corrector solo.

El diccionario flexivo, posee una velocidad de respuesta del orden de 86 mil palabras por segundo en promedio, éste es crucial en el proceso de corrección y aumentar su velocidad implica aumentar linealmente la de todo el corrector, sin embargo el espacio de memoria que se necesitaría, si se tratase de usar para el total de las transformaciones morfológicas posibles y representadas en el diccionario es inviable.

Sin embargo se puede aumentar la velocidad del diccionario flexivo interno otro orden de magnitud, mediante el uso de filtros de *Bloom* [79] en el diccionario de prueba; por supuesto requiriendo mayor uso de memoria, bajando el tiempo de aceptación de las palabras más comunes. Los filtros de *Bloom* se pueden preprocesar y almacenar, mejorando notablemente la velocidad sin penalidad alguna.

Se recuerda que se ha implementado una versión de caché activo por cada palabra, el cual llevó el al corrector a la velocidad actual medida, siendo anteriormente 5 a 10 veces más lento. Si se implementase un filtro de Blum, debiese desactivarse este último caché ya que varios caché en cascada por lo general no mejoran sino hacen todo lo contrario.

Aplicación en Otros Idiomas

Con otros idiomas, es de prever que los métodos propuestos se comportarán igualmente bien, si se aplicasen a idiomas de similar complejidad flexiva y estructura morfológica. La experiencia de expansión del ***Aspell*** a más de 50 idiomas, confirma esta regla.

En este caso, para poder replicar este corrector en otros idiomas, puede resultar útil generar herramientas de edición específicas en función de los formatos de reglas morfológicas para permitir incluir las características lingüísticas de los otros idiomas.

Son candidatos preferenciales los idiomas indoeuropeos aglutinantes que resulten similarmente flexivos como ser: *portugués, francés, polaco, alemán, holandés, italiano, románico, noruego, danés, checo, ruso, gallego, catalán, etc.*

Extracción de Significados

Tal vez resulte útil agregar información específica en los diccionarios para usarla en sistemas de diálogo, con inteligencia artificial lingüística y aprendizaje automático. Esto es posible debido a que el sistema ya permite, durante el análisis de la flexión y derivación morfológica, obtener fácilmente rica información ontológica y semántica extraída de las palabras, a condición de que estén presentes en los diccionarios.

Corrección Multipalabra

Sería deseable mejorar la calidad de la corrección, usando contexto multipalabra de manera que el sistema futuro provea información útil, recurriendo al contexto cercano. Esto sin duda proveerá mejoras en la corrección de errores y en el reconocimiento de entidades nombradas complejas (*NER*), siendo esto, hoy en día un tema abierto.

Corrección usando Contexto Gramatical

Existe la posibilidad de realizar modelos para obtener corrección de errores de mucho más alto nivel, como ser los de concordancia sujeto-verbo, o verbo-complementos (*directo/indirecto/agente*) o de los circunstanciales con los tiempos verbales.

Del mismo modo se puede encarar la corrección de casos más sofisticados, como cuando es necesaria la concordancia entre tipo de verbo y la viabilidad de uso de determinada palabra por su contenido semántico como núcleo de algún complemento, incluyendo atribuciones de sentido común, por ejemplo: “comer ideas” que solo se puede entender como figurativo.

Este tipo de correcciones implican un análisis gramatical y semántico que trasciende la escritura regular. Su operación puede invadir fácilmente el terreno de las licencias ‘poéticas’ de los escritores y por ende, estimamos que solamente sería conveniente que brinden “sugerencias”. Se ha observado que el corrector de estilo del procesador de texto MS-Word hace algo de esto, de título muy básico, pero que sin duda resulta útil a la hora de corregir concordancias que se le puedan escapar a un redactor avezado.

Recientemente aparecieron productos como ***www.grammarly.com***, se ofrecen como servicios en la nube, y por ahora son exclusivamente para el inglés, no español; estimamos por la dificultad con la que tropezamos: su gran cantidad de flexiones.

Terminología Científica

Tal vez la aplicación potencialmente más interesante y controvertida es la de utilizar este corrector para complementarlo con un "sugeridor" de textos basado en predicción inteligente, en tiempo real. Sería útil para terminologías específicas, como la legal, la médica y la científica en general, mejorando la calidad de la escritura y asistiendo desde un punto de vista tecnológico pero humano a la vez, por el carácter de las sugerencias.

Un sistema así permitiría a los profesionales escribir terminologías complejas sin riesgo de error y con una ayuda memoria; por ejemplo: *"gingivitis ulcerosa"*; en este caso el sistema presentado, dada su alta velocidad, podría, mientras se escribe la segunda palabra, sugerir correctamente el "*siguiente término completo*" incluso correctamente flexionado, conforme a la concordancia del caso. Esto, teniendo en cuenta el tipo de documento y el sesgo del profesional, que en este caso estaría llenando una ficha de una historia clínica médica electrónica, *(H.C.M.)*.

En este sentido, las aplicaciones de este trabajo se pueden extender a numerosos campos semejantes de aplicación, mejorando la calidad del material ingresado por las personas a las máquinas, y posibilitando un mejor análisis de datos ya consignados, abriendo un mundo de posibilidades de análisis automatizado de los textos para mejora de la calidad de la ciencia aplicable, continuando con el ejemplo de medicina, a los diagnósticos preventivos, las interacciones medicamentosas, por solo mencionar uno de los rubros más sensibles de uso. Es de notar que la tasa actual de errores ortográficos en el ingreso de texto en las HCM interfiere y hasta inhabilita procesos de análisis automático, prácticamente impidiendo su uso, o tornando poco confiables los resultados.

Con la posibilidad de reconstrucción ortográfica inteligente, emulando la corrección realizada por una persona real, se podrá sin dudas lograr una mejora significativa en calidad de todos los procesos automatizados, especialmente en los rubros en donde procesar lenguaje resulte de utilidad. §

Bibliografía

[1] Lawrence Phillips “Hanging on the Metaphone”, Computer Language, vol. 7, no. 12, pp. 39-43, 1990.

[2] Santana, O.; Carreras, F.; Pérez, J.; Rodríguez, G. “Relaciones morfoléxicas prefijales del español” Boletín de Lingüística, Vol. 22. ISSN: 0798-9709. Jul/Dic, 2004. 79/123.

[3] Real Academia Española “Diccionario de la Lengua Española, 22ª edición”, http://buscon.rae.es/draeI/, 2/2012

[4] Santana, O.; Pérez, J.; Carreras, F.; Duque, J.; Hernández, Z.; Rodríguez, G. “FLANOM: Flexionador y lematizador automático de formas nominales. Lingüística Española Actual” XXI, 2, 1999. Ed. Arco/Libros, S.L. 253/297

[5] "TST -Ternary Search Tree " http://www.nist.gov/dads/HTML/ternarySearchTree.html, 05/2012

[6] Mehlhorn, K. Dynamic Binary Search. SIAM Journal on Computing 8, 2 (May 1979), 175-198.

[7] “ASPELL”, http://aspell.sourceforge.net/man-html/Affix-Compression.html

[8] Chomsky, N., Aspects of the Theory of Syntax, MIT Press, Cambridge, Mass., 1965. Noam Chomsky “The Noam Chomsky Website”, http://www.chomsky.info/, 6/2011

[9] Francisco Escoleano, Miguel Angel Cazorla, María Isabel Alfonso, Otto Colomina, Miguel Ángel Lozano, “Inteligencia Artificial Modelos Técnicas y Áreas de Aplicación”, Thompson Editores Spain Paraninfo s.a., 2003, ISBN 84-9732-183-9

[10] “GATE: General Architecture for Text Engineering, http://gate.ac.uk/, 6/2011

[11] “JLEX: Automatic Lexer Generator for Java”, www.cs.princeton.edu/~appel/modern/java/JLex/, 6/2011

[12] Santana, O.; Carreras, F.; Pérez, J.; Rodríguez, J. "Parasynthetic morpholexical relationships of the Spanish: lexical search beyond the lexicographical regularity" Proceedings of the IADIS International Conference. Applied Computing. 2006. ISBN: 972-8924-09-7. Febrero, 2006. 627/631. http://acceda.ulpgc.es/bitstream/10553/471/1/3689.pdf , 06/2011

[13] Yarowsky D. (publicaciones) http://www.cs.jhu.edu/~yarowsky/pubs.html, 06/2011

[14] Reynaert M., ‘Proceedings of the 20th international conference on Computational Linguistics’ 2004 (COLING 2004) Geneva (pp. 834-840)

[15] Freeling http://nlp.lsi.upc.edu/freeling/ , 06/2011

[16] Hohendahl, A. T.; Zanutto, B. S.; Wainselboim, A. J. “Desarrollo de un algoritmo para la medición del grado de similitud fonológica entre formas escritas” SLAN2007. X Congreso Latinoamericano de Neuropsicología 2007, Buenos Aires, Argentina

[17] Hohendahl A. T., Zelasco J. F., "Lematizador Morfosintáctico y Semántico Robusto con Flexionador y Estimador Idiomático, usando algoritmos eficientes y compactos para idiomas muy ricos como el español"
XII Congreso Argentino de Ciencias de la Computación, Potrero de los Funes, San Luis, Argentina. CACIC 2006 - ISBN 950-609-050-5

[19] Hohendahl, A. T. "Procesamiento de Lenguaje Natural Robusto" ProLen 2011, Primer Encuentro de Grupos de Investigación sobre Procesamiento del Lenguaje, UBA, Facultad de Filosofía y Letras, del 4-6 de mayo 2011, Biblioteca Nacional, CABA, Argentina. http://web.fi.uba.ar/~ahohenda/docs/prolen_robust.pdf , 01/2013

[20] Hohendahl, A. T. "Plataforma para Desarrollo de Agentes Inteligentes" ProLen 2011, Primer Encuentro de Grupos de Investigación sobre Procesamiento del Lenguaje, UBA, Facultad de Filosofía y Letras, del 4-6 de mayo 2011, Biblioteca Nacional, CABA, Argentina.
http://web.fi.uba.ar/~ahohenda/docs/prolen_agents.pdf, 01/2013

[21] Hohendahl, A. T. | Zelasco, J. F. "Algoritmos eficientes para detección temprana de errores y clasificación idiomática para uso en procesamiento de lenguaje natural y texto"
WICC2006 - VIII Workshop de Investigadores en Ciencias de la Computación, Universidad de Morón, 2006, ISBN: 950-9474-35-5

[22] Vapnik V., Golowich S., and Smola A. "Support vector method for function approximation, regression estimation, and signal processing" 1997. In: Mozer M.C., Jordan M.I., and Petsche T. (Eds.) Advances in Neural Information Processing Systems 9, MA, MIT Press, Cambridge. pp. 281-287.

[23] Sutton, C., McCallum, A. "An Introduction to Conditional Random Fields for Relational Learning." In "Introduction to Statistical Relational Learning". Edited by Lise Getoor and Ben Taskar. MIT Press. (2006).

[24] Minorthird - http://minorthird.sourceforge.net/

[25] MALLET for Java - http://mallet.cs.umass.edu/

[26] HCRF library (including CRF and LDCRF) For C++ and Matlab - http://sourceforge.net/projects/hcrf/

[27] CRF++ for C++ - http://crfpp.sourceforge.net/

[28] CRFSuite for C++ - http://www.chokkan.org/software/crfsuite/

[29] Sunita Sarawagi's CRF package for Java - http://crf.sourceforge.net/

[30] Armenta, A., Escalada, J. G., Garrido, J. M. & Rodríguez, M. A. (2003). "Desarrollo de un corrector ortográfico para aplicaciones de conversión texto-voz." In Procesamiento del Lenguaje Natural, 31, pp. 65-72.

[31] Shannon, Claude. "Communication in the presence of noise" 1949 Proceedings of the IRE.-37(1):10-21.

[32] A. J. Viterbi. "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm" IEEE Transactions on Information Theory, pag 260-269, Abril 1967.

[33] Lluís Padró, Cirera, Montserrat "Comparing methods for language identification". Procesamiento del lenguaje natural. Nº 33 (septiembre 2004), pp. 155-161

[34] Gurlekian, J. A., Colantoni, L. y Torres, "El alfabeto fonético SAMPA y el diseño de corpora fonéticamente balanceados". H. Fonoaudiológica. Editorial: ASALFA. Tomo: 47, Numero: 3, pp 58-69, Diciembre 2001

[35] Brad Merrill, "CsLex: a lexical analyzer generator for C# " Microsoft, 1999 http://www.cybercom.net/~zbrad/DotNet/Lex/Lex.htm, 4/2013

[36] Grodzinsky, Y. "Imaging the Grammatical Brain" 2003. In M. Arbib, ed., Handbook of Brain Theory and Neural Networks, 2nd edition, 551-556. Cambridge, MA: MIT Press.

[37] Vintsyuk, T.K. "Speech discrimination by dynamic programming". Kibernetika, Vol. 4, pp. 81–88, Jan.-Feb. 1968.
https://en.wikipedia.org/wiki/Dynamic_time_warping

[38] Scott Sadowsky & Ricardo Martínez Gamboa. "LIFCACH 2.0: Lista de Frecuencias de Palabras del Castellano de Chile" http://sadowsky.cl/lifcach-es.html

[39] "TRIE" http://www.cs.bu.edu/teaching/c/tree/trie/ 01/2013

[40] Freund, Yoav; Schapire, Robert E. "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting." (1995). CiteSeerX: 10.1.1.56.9855

[41] Brigitte van Berkel and Koenraad De Smedt, "Triphone Analysis: A combined method for the correction of Orthographical and Typographical Errors." Proc. of the Second Conference on Applied Natural Language Processing, Austin TX, 9-12 Feb. 1988 (pp. 77-83).

[42] Levenshtein VI "Binary codes capable of correcting deletions, insertions, and reversals". (1966). Soviet Physics Doklady 10: 707–10.

[43] "The Development of Proof Theory" Stanford Encyclopedia of Phylosophy - Apr 2008 http://plato.stanford.edu/archives/fall2008/entries/proof-theory-development/

[44] Freund, Yoav; Schapire, Robert E. (1995). "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting". CiteSeerX: 10.1.1.56.9855.

[45] T. Zhang, "Statistical behavior and consistency of classification methods based on convex risk minimization", Annals of Statistics 32 (1), pp. 56-85, 2004.

[46] Peterson, J.L. (1980) "Computer programs for detecting and correcting spelling errors". Communications of ACM, 23, pp. 676-687.

[47] Damerau, F.J. (1964) "A technique for computer detection and correction of spelling errors." Communications of ACM, Vol. 7, No. 3, pp. 171-177, March, 1964.

[48] Angell, R.C., Freund, G.E. & Willett, P. (1983) Automatic spelling correction using a trigram similarity measure. Information Processing & Management, 19, 255-261

[49] Daelemans, W., Bakker, D. & Schotel, H. (1984) "Automatische detectie en correctie van spelfouten." Informatie, Vol. 26, pp. 949-1024.

[50] Kukich, Karen. 1992. Techniques for automatically correcting words in text. ACM Computing Surveys Bol 24. Nº 4, December 1992. pp. 377-437.

[51] Jurafsky, Dan and James H. Martin. 2000. Speech and Language Processing. Prentice-Hall. Chapter 5.

[52] Peter Norvig. 2007. How to Write a Spelling Corrector. On http://norvig.com.

[53] Carpenter, G.A. & Grossberg, S. (2003), Adaptive Resonance Theory, In Michael A. Arbib (Ed.), "The Handbook of Brain Theory and Neural Networks, Second Edition" (pp. 87-90). Cambridge, MA: MIT Press

[54] Robert C. Russell and Margaret K. Odell; US patent 1261167, issued on 1918-04-02 (Archived) y US patent 1435663 issued on 1922-11-14 (Archived)

[55] Jordi Atserias, Maria Fuentes, Rogelio Nazar, Irene Renau "Spell-Checking in Spanish: The Case of Diacritic Accents" 2012 Proceedings LREC, IStambul, Turkey. ELRA isbn: 978-2-9517408-7-7

[56] Kukich, K., "A comparison of some novel and traditional lexical distance metrics for spelling correction", lNNC-90-Paris, pp. 309-313 (1990)

[57] Alberga, C.N. String Similarity and Misspelling, In Communications of ACM, Vol. 10, No. 5, pp. 302-313, May, 1967.

[58] Victoria J. Hodge and Jim Austin "A Comparison of Standard Spell Checking Algorithms and a Novel Binary Neural Approach" IEEE transactions on knowledge and data engineering, Vol.15. Nº5 Sep/OCT 2003.

[59] Yang Zhang1, Pilian He1, Wei Xiang2, Mu Li "Discriminative Reranking for Spelling Correction" PACLIC 20 / Wuhan, China 1-3 Nov 2006.

[60] Kernighan et. al. "A Spelling Correction Program Based on Noisy Channel Model", In Proceedings of COLING-90, The 13th International Conference On Computational Linguistics, Vol 2. 1990

[61] Stanier, Alan, "How accurate is Soundex matching", Computers in Genealogy Vol. 3, No. 7, pp. 286-288. September 1990.

[62] Christian, Peter, "Soundex – can it be improved?", Computers in Genealogy Vol. 6, No. 5, March, 1998.

[63] Brill, E. and Moore, R. C. An Improved Error Model for Noisy Channel Spelling Correction. In proceedings of 38th Annual meeting of Association for Computational Linguistics, pp. 286-293, 2000.

[64] Turba, T. N. 1981. Checking for spelling and typographical errors in computer based text. SIGPLANSIGOA Newsletter (June), pp/ 51-60

[65] Peterson, L.J. A Note on Undetected Typing Errors. In Communications of ACM, Vol. 29, No. 7, pp. 633-637, July, 1986.

[66] Sproat, R et. Al. A stochastic finite-state word-segmentation algorithm for Chinese Volume 22 , Issue 3 (September 1996) table of contents, p 377 – 404, 1996, MIT Press

[67] Toutanova, K. and Moore, R. C. "Pronunciation Modeling for Improved Spelling Correction", In proceedings of 40th Annual meeting of Association for Computational Linguistics, July 2002, pp. 144-151.

[69] Peter D. Turney, Patrick Pantel "From Frequency to Meaning: Vector Space Models of Semantics" Journal of Artifcial Intelligence Research 37 (2010) 141-188

[70] Salton, G., Wong, A. y Yang, C. S. "A vector space model for automatic indexing." (1975) Communications ACM, 18(11) pp. 613-620

[71] Roger Mitton "Spellchecking by computer" Journal of the Simplified Spelling Society, Vol 20, No 1, 1996, pp 4-11

[72] Johannes C. Ziegler, Jonathan Grainger, Marc Brysbaert "Modelling word recognition and reading aloud", European Journal Of Cognitive Psychology 2010, 22 (5), pp. 641-649

[73] Pollock, J. J. & Zamora, A. (1984). "Automatic spelling correction in scientific and scholarly text." In Communications of the ACM, 27 (4), pp. 358-368.

[74] Luz Rellos & Baeza-Yatess R. "The presence of English and Spanish dyslexia in the Web" New Review of Hypermedia and Multimedia, 2012, pp. 1-28

[75] Flora Ramírez Bustamante, Enrique López Díaz. “Spelling Error Patterns in Spanish for Word Processing Applications” Microsoft press, 2004

[76] Chris Manning and Hinrich Schütze, “Foundations of Statistical Natural Language Processing”, MIT Press. Cambridge, MA: May 1999.

[77] Royal Skousen “Analogical Modeling of Language”. Dordrecht: Kluwer Academic Publishers. XII+ pp. 212 ISBN 0-7923-0517-5. 1989

[78] Kann, Viggo et. al., “Implementation Aspects And Applications Of A Spelling Correction Algorithm”, May 1998.

[79] B. H. Bloom. “Space/time trade-offs in hash coding with allowable errors.” Communications of the ACM, 13(7) pp. 422–426, 1970.

[80] Dempster, A. P., N. M. Laird, and D. B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm” 1977 - Journal of the Royal Statistical Society, 39(B).

[81] Yarowsky, D. “Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French.” In Proceedings. ACL. 1994

[82] Sebastian Deorowicz, Marcin G. Ciura “Correcting Spelling Errors By Modeling Their Causes” Int. J. Appl. Math. Comput. Sci., 2005, Vol. 15, No. 2, 275–285

[83] Antonio Frías Delgado “Distribución de Frecuencias de la Longitud de las palabras en español: Aspectos Diacrónicos y de Estilometría” I Congreso Internacional de Lingüística de Corpus (CILC’09) Universidad de Murcia 7-9 de mayo, 2009

[84] J. Pedler. “Computer Correction of Real-word Spelling Errors in Dyslexic Text” PhD thesis, Birkbeck, London University, 2007.

[85] Mitton, Roger (2010) “Fifty years of spellchecking”. Writing Systems Research 2 (1), pp. 1-7. ISSN 1758-6801

[86] Martínez, L., Herrera, C., Valle, J., & Vásquez, M. (2011). Memoria de Trabajo Fonológica en Preescolares con Trastorno Específico del Lenguaje Expresivo. Psykhe, 12(2).

Bibliografías útiles y relacionadas, aunque no explícitamente indicadas.

Erikson, K. Approximate Swedish Name Matching – Survey and Test of Different Algorithms, 1997.

Holmes, David and McCabe, M. C., Improving precision and recall for Soundex

Ren, X. & Perrault, F. (1992). The typology of unknown words: an experimental study of two corpora. In Proceedings of Coling, pp. 408-414.

Veronis, J. (1988). Morphosyntactic correction in natural language interfaces. In Proceedings of the 12th conference on Computational linguistics, pp. 708-713.

Yannakoudakis, E. & Fawthrop, J. D. (1983). The rules of spelling errors. In Information Processing & Management, 19 (2), pp. 87-99.

Luz Rello1; Ricardo Baeza-Yates, Horacio Saggion, Jennifer Pedler
"A First Approach to the Creation of a Spanish Corpus of Dyslexic Texts"

Apéndices y Anexos

En esta sección se han condensado algunas cuestiones que se consideran importantes pero no centrales al trabajo, y pueden arrojar algunas luces sobre numerosos temas; en especial en lo que respecta a un análisis considerando el trabajo de *Claude Shannon*. En la sección posterior, se trata un modelo de espacio vectorial de palabras que presenta una visión de cómo se puede interpretar el comportamiento de un sistema tan complejo como un corrector ortográfico en un universo descripto matemáticamente, mostrando similitud entra los conceptos geométricos y las operaciones de los algoritmos; ayudando a comprender intuitivamente las acciones.

Análisis de la Ganancia de Calidad

Según el trabajo de Shannon de 1948, la cantidad de información transmitida dada una codificación en un canal ruidoso se puede medir con la entropía.

Si a la probabilidad de que una palabra llegue con un error de ortografía la llamamos: P_E y la probabilidad de que llegue sin error sería P_D, entonces la entropía E_M sería:

$$E_M = -P_E \times Log_2(P_E) - P_D \times Log_2(P_D)$$

Si un texto llega con error, y este error es un determinado porcentaje del mismo, considerada como una probabilidad de error P_E, podríamos decir que la capacidad del canal es menor, dado el 'ruido' y su entropía E_E se calcularía así:

$$E_E = -P_E \times Log_2(P_E) - (1 - P_E) \times Log_2(1 - P_E)$$

Si luego de recibido, a toda palabra detectada con error, se le aplica un corrector, y éste corrige una determinada fracción F_C de las mismas, la entropía del mensaje corregido E_C sería.

$$E_C = -P_E \times F_C \times Log_2(P_E \times F_C) - (1 - P_E \times F_C) \times Log_2(1 - P_E \times F_C)$$

Resultando una ganancia de información que podríamos poner como diferencia de las entropías $E_T - E_C$ en este tratamiento de señal.

$$E_T - E_C$$

Análisis Estadístico Comparativo

Si se toma cada letra y sus frecuencias de aparición en corpus, como estimadores de su probabilidad de aparición, la entropía calculada resulta de 4.12 bits por letra, para el español. Notablemente se ha realizado el cálculo de la entropía de los sets de pruebas y en el caso del set artificial de 10k letras, ésta resulta 4,273 corroborando que se trata de una distribución de similares características que el idioma del que se parte.

Como las palabras del idioma, contienen un número variable de letras, se puede estimar el número promedio de bits por palabra, basándose en su distribución promedio.

Estimaremos la cantidad de información si usamos las distribuciones vistas a lo largo de este trabajo, para el español tomando ~7.5 letras como promedio, resulta ~31 bits por palabra aproximadamente.

Es decir, en español habría 2^{31} símbolos diferentes, equivalentes a 2.15×10^{9} palabras, cifra que está en el orden de estimación de las predicciones morfológicas y flexivas del idioma dadas por numerosos lingüistas, condiciendo con la teoría de *Shannon*.

Por otro lado dado lo ralo del espacio de palabras válidas de un idioma, respecto al de combinaciones de letras, la relación señal a ruido de este canal, analizado como un canal ruidoso, es muy grande, permitiendo una compacta codificación mediante reglas de morfología y fonética, que es el lenguaje mismo.

Existe una posibilidad de analizar la cantidad de información que contienen las palabras de un mensaje en determinado idioma, conociendo la probabilidad de aparición de cada palabra y realizando la suma de las entropías aportadas por cada una. Si bien este modelo es usado en procesamiento de lenguaje, no refleja la verdadera información contenida en un mensaje pues al considerar la entropía de palabras sueltas, descartamos de lleno la información de posición (sintaxis) y esto rompe con el modelo semántico, pues la desambiguación de palabras se realiza por contexto fuertemente posicional, en idiomas como el español, no tanto en el alemán y menos aún en el latín, porque esta información está contenida en la declinación. En consecuencia no se profundizó más en este trabajo, este tipo de estudio teórico que pretende asociar los fragmentos de texto a trozos de información pura, estadísticamente descripta en su totalidad.

A continuación presentamos un modelo estadístico interesante derivado de uno que ha dado resultados asombrosos en minería de texto, es el modelo de "espacio vectorial de palabras" (del inglés *VSM: Vector Space/ Semantic Models*) y vale la pena analizarlo para entender bien el delicado trabajo que debe hacer un corrector de ortografía que se jacte de 'bueno' y parecido a lo que una persona es capaz de realizar.

Palabras como un Espacio Vectorial

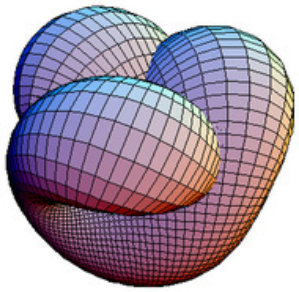
Si se define una métrica entre formas escritas, y se logra que ésta cumpla las premisas de una distancia (es positiva y satisface la desigualdad triangular; en caso más general, *Cauchy-Bunyakovski-Swartz,*) ya se estaría definiendo un espacio vectorial.

Entonces se pueden tomar las palabras de un idioma como puntos de un espacio vectorial usando esa métrica. Surge la inminente pregunta de cual debiesen de ser las bases del mismo. Estas bases por ejemplo podrían ser funciones de las letras, la longitud en letras, características fonéticas de los grupos formados por su secuencia, características de las sílabas, u otro cálculo cualquiera; siempre que se cumplan las premisas de poder calcular las distancias entre dos palabras a partir de ellas. La naturaleza misma de las bases no es importante, inclusive no es necesario que sean continuas o finitas; mientras se cumplan con las premisas antedichas.

Si este espacio poseyera más de 3 dimensiones, una representación de la totalidad de las palabras resultaría imposible, sin embargo conceptos útiles como similitud, márgenes y caminos entre todo punto, y un cluster son fácilmente interpretables.

En este espacio existirían sin dudas sectores densos, conteniendo todas las posibles palabras y si las bases están bien escogidas, éste será formado por compactos grupos aglutinados, a pesar de que los símbolos son discretos. A estos sectores densos o clusters poblados de palabras, lo llamaremos ***manifold*** haciendo referencia a una zona compacta del espacio vectorial mencionado.

A este efecto citamos un ejemplo (*obtenido de Wikipedia*) que muestra claramente este tipo de zonas o subespacios, si tan solo hubiese 3 dimensiones.



La forma de este ***manifold*** seguramente no será predecible y se estima difícilmente representable, pero no resulta arduo de imaginar su comportamiento, en especial las palabras con errores debieran de estar ‘cercanas’ al ***manifold***.

Ahora, podemos definir una relación de llenado como cociente entre el número posible de formas escritas, como el tamaño total del espacio, y el subespacio de las formas que realmente existen y son palabras en un idioma.

Si definimos como deseables a las palabras, las no palabras serán ruido, y la relación entre el número de no-palabras posibles y el de palabras se podría interpretar como una

relación señal a ruido de ese espacio. Es decir el ruido allí, es toda manifestación de una forma escrita fuera del ***manifold*** donde las palabras existen.

Cuanto mayor será la relación señal a ruido, tanto más fácil será detectar y corregir errores de caracteres, considerando la tarea de corrección como el hallar los caminos más cortos desde la palabra errada *fuera del* ***manifold*** hacia el interior del mismo, donde se halla la palabra correcta.

El estudio hecho en la **sección 6.5** nos habla de una relación de 10^{34} entre el total de puntos de este espacio, tomado como discreto y el número de puntos ocupados por palabras reales o posibles. Esto indica que la probabilidad de que una palabra escogida aleatoriamente pertenezca a ese subespacio es muy baja, ciertamente 10^{-34}, lo cual permitiría estimar la entropía total del conjunto de palabras existentes. Esta entropía resultaría muy baja, si la tomamos como un solo punto, - p $\log_2(p)$ y resulta 1.09 10^{-31}. En consecuencia es un espacio muy denso y el poder disminuir la entropía de una palabra cualquiera debiera de ser posible. Esto condice , lo que no podría ser de otro modo, con la noción de distancia y la de señal a ruido.

Si las bases de representación están bien escogidas, pudiendo ser obtenidas por algún mecanismo relacionado con operaciones morfológicas como flexión y derivación, tanto más compacta será la representación y más sencilla una operación de corrección.

Lo interesante es imaginar estas tres zonas, con las palabras raíz, formando un esqueleto denso de este subespacio, como un ***manifold*** interno, rodeado de una zona externa, representando la flexión con las reglas morfológicas y de derivación.

Este es el caso de los correctores como el presentado, que ante errores simples, permiten corregir con asombrosa precisión, llegando al 99%, partiendo solamente de un esqueleto central, pues solo contienen las formas raíz de las palabras.

Entonces, un corrector, mejora la relación señal a ruido o reduce la entropía en este espacio, lo cual es equivalente. Así se obtienen reducciones de error cercanas al 99% que implican algo así como 40dB de mejora en la relación señal a ruido.

Este tipo de ganancias de calidad es por cierto difícil de conseguir en un sistema electrónico analógico de procesamiento de señales, salvo casos muy especiales.

Precisamente, los sistemas usados para codificar señales desde satélites y sondas enviadas al espacio profundo como el ***Voyager***, a costa de transmitir lento y con códigos redundantes, usan algoritmos muy robustos como los de *Reed-Solomon*, para lograr decodificar símbolos con ganancias de relación señal a ruido de este orden.

Observación

Es notable que la inteligencia humana de manera no deliberada y nativa, haya logrado una representación tan ingeniosa y compacta al codificar robustamente las palabras con longitud variable mediante mecanismos cognitivos con miles de reglas de formación morfológicas y fonéticas. Resulta curiosamente óptima la eficiencia de transmisión de las ideas contenidas en esas palabras, con una economía importante en el número de

símbolos, minimizando el número de bits transmitidos, o cantidad de información bruta. En caso de ser hablados, representados físicamente como fonemas, como resultado de esfuerzos mecánicos de nuestro aparato fono-articulador, se ha minimizando la energía usada en crear y transmitir los mensajes, lo más robustamente posible.

Asimismo la capacidad de corrección tanto fonética, ortográfica como semántica, y contextual, que poseen las personas, es tan asombrosa que supera considerablemente la lograda hoy en día mediante sistemas automatizados como el presentado aquí, además de tantos otros.

Sin embargo no es de extrañar este comportamiento optimizado ya que la biología siempre ha economizado recursos y en consecuencia minimizado la entropía de todo proceso, incluida la codificación del lenguaje escrito, lo que permite una comunicación robusta conjuntamente con un proceso de corrección de errores muy exitoso.

Apéndice 1

Etiquetas del Tokenizador

Tipos de 'etiquetas' utilizadas en el Lexer inicial (Tokenizador). La sintaxis es de C# y el comentario de cada tipo está en verde.

```
public enum LexType
{
    Unknown = 0, // tipo desconcido

    SPACE = 1, // espacio

    // Alpha-Num's
    UnkAnum = 10, // Alfanumerico desconocido
    UNDERSCORE = 11, // guión bajo: _

    Time = 20, // Tiempo o Fecha

    Word = 30,  // Palabra Alfabética
    Tweet = 31,  // Nombre de Twitter
    Hashtag = 32, // Hashtag de Twitter

    Number = 40, // Número (decimal)
    Hex = 41, // Número (hexadecimal)
    Fraction = 42, // Número Fraccional

    Phone = 60, // Teléfono
    Email = 61, // Dirección de correo electrónico
    Url = 62, // Dirección de internet
    Uri = 63, // Dirección de internet
    Http = 64, // Dirección de internet
    Money = 65, // Expresión de Moneda
    Chem = 66, // Expresión Química
    Math = 67, // Expresión Matemática

    // chars/groups working as chars
    COMMA = 91,
    SEMIC = 92,
    ELLIPSES = 93,
    DOT = 94,
    DDOT = 95,
    TILDE = 96,
    DTILDE = 97,
    EQUAL = 98,
    PLUS = 99,
    UMINUS = 100,
    MINUS = 101,
    TIMES = 102,
    DIVIDE = 103,
    MOD = 104,  // Módulus = Percent
    POWER = 105, // ^
    RSHIFT = 106, // >>
    LSHIFT = 107, // <<
    LPAREN = 108, // (
    RPAREN = 109, // )
    LCBRACK = 110, // {
```

```
    RCBRACK = 111, // }
    LBRACK = 112, // [
    RBRACK = 113, // ]
    QUEST = 114, // ? ¿
    EXCL = 115, // ^^
    GT = 116, // >
    LT = 117, // <
    NEQ = 118, // !=
    GTEQ = 119, // >=
    LTEQ = 120, // <=
    SQ = 121, // ²
    CUB = 122, // ³
    NUMERAL = 123, // #
    PERMILL = 124, // per milar 0/00
    PLUSMINUS = 125, // +/-
    ONE = 126, // 1 como superindice
    SQRT = 127, // Raiz Cuadrada (símbolo)
    BACKSLASH = 128, // \
    OR = 129, // |
    AND = 130, // &
    XOR = 131, // ^
    AT = 132,    // @ Ampersand
    APROX = 133, // ≈ aprox. igual
    SIMIL = 134, // ~ similar
    EQUIV = 135, // ≡ Equivalente
    SUM = 136, // Sumatoria
    DELTA = 137, // Delta
    DERIVE = 138, // Derivada parcial
    INFINIT = 139, // infinito
    NOT = 140, // Not (inicial)
    INTEGRAL = 141, // Integral
    INTERSEC = 142, // Intersección
    MATHSYM = 143, // Other Math Symbols
    PUNCT = 144,   // OTHER Punctuation
    PRODUCT = 145, // Productorial
    OMEGA = 146,
    MICRO = 147,

    SECC = 167,     // § Sección Alt+0167
    GRAD = 176,     // ° Grado Alt+0176
    VINIETA = 149,  // • Viñeta Alt+0149

    // for chars
    UNKNOWN = 200, // carácter no contemplado

    MORE = 666, // contiuación, uso interno..

    EOP = 998, // ¶ End of Paragraph
    EOF = 999, //   End of File
}
```

Gramática del Lexer

La Gramática para el LexCS utilizada (parte) tiene este formato:

```
DIGIT =[0-9]
NUMBER ={DIGIT}*[.,]?{DIGIT}+([eE][-+]?{DIGIT}+)?
SPC =[ \t\r\n\f\012]
INT = [?¿]
EXC = [!¡]
VL = [a-zA-Z]
VC = [~0-9a-zA-Z_-]
VX = [0-9a-zA-Z_-]
VS = ({VC}+\.{VC}+)
VA = ({VC}+(\.)?{VC}+)+
VP = ([/\\]?{VC}*)
VQ = ([0-9a-zA-Z_\-?&%@=!~#+\.\\\*:,;<>]*)
VD = (%{DIGIT}{DIGIT})
AW = {VP}*{VQ}*
UN = (({VC}+|{VC}+@{VC}+):?)
VW = ({VC}*(\.{VC}+))
ALPHA = [^\]<>[(\.,\";:?!¿¡«»'``´)^@~|}\{_/\\+=&$#*%-" "]
LETTER = [a-zA-ZáãàäÄÀÁÃèéëÈÉËìíïÌÍÍòóõöÒÓÖÕûùúüÛÜÙÚñÑçÇ]
LETTEX = [a-zA-ZáãàäÄÀÁÃèéëÈÉËìíïÌÍÍòóõöÒÓÖÕûùúüÛÜÙÚñÑçÇ_0-9]
SYMB = [0-9º¹²³½¾¼⅓⅔⅛⅜⅝⅞_']
PRE = [º]
APO = [']
SUF = [º¹²³]
TIME = (([01]?[0-9]|2[0-4])\s*:\s*[0-5]?[0-9])
WORD = {PRE}?{LETTER}+(-{LETTER}+)*({APO}?{LETTEX}+)*
// PHONE = ([-+ /#]*(15|{DIGIT}{DIGIT}+|\({DIGIT}+}\)))([-+
/#]*(15|{DIGIT}{DIGIT}{DIGIT}+|\({DIGIT}+}\)))+
EMAIL = {VA}@{VS}+
//14-10-2010 from: http://data.iana.org/TLD/tlds-alpha-by-domain.txt
IANA =
\.(ac|ad|ae|aero|af|ag|ai|al|am|an|ao|aq|ar|arpa|as|asia|at|au|aw|ax|a
z|ba|bb|bd|be|bf|bg|bh|bi|biz|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|cat|
cc|cd|cf|cg|ch|ci|ck|cl|cm|cn|co|com|coop|cr|cu|cv|cx|cy|cz|de|dj|dk|d
m|do|dz|ec|edu|ee|eg|er|es|et|eu|fi|fj|fk|fm|fo|fr|ga|gb|gd|ge|gf|gg|g
h|gi|gl|gm|gn|gov|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|hr|ht|hu|id|ie|il|i
m|in|info|int|io|iq|ir|is|it|je|jm|jo|jobs|jp|ke|kg|kh|ki|km|kn|kp|kr|
kw|ky|kz|la|lb|lc|li|lk|lr|ls|lt|lu|lv|ly|ma|mc|md|me|mg|mh|mil|mk|ml|
mm|mn|mo|mobi|mp|mq|mr|ms|mt|mu|museum|mv|mw|mx|my|mz|na|name|nc|ne|ne
t|nf|ng|ni|nl|no|np|nr|nu|nz|om|org|pa|pe|pf|pg|ph|pk|pl|pm|pn|pr|pro|
ps|pt|pw|py|qa|re|ro|rs|ru|rw|sa|sb|sc|sd|se|sg|sh|si|sj|sk|sl|sm|sn|s
o|sr|st|su|sv|sy|sz|tc|td|tel|tf|tg|th|tj|tk|tl|tm|tn|to|tp|tr|travel|
tt|tv|tw|tz|ua|ug|uk|us|uy|uz|va|vc|ve|vg|vi|vn|vu|wf|ws|ye|yt|za|zm|z
w)
URI = {UN}*{VW}*{IANA}{AW}
URL = ({VL}+://)?{URI}
HTTP = https?:(//)?{URI}
HEXA = (0x)[0-9A-Fa-f]+
ANUM = {LETTEX}+
QUOTES = (``|<<|>>|'')
// chemical's
NAT = ([1-9][0-9]*|n)
...
```

Apéndice 2

ETIQUETAS EAGLES (v. 2.0)

El sistema desarrollado utiliza internamente un conjunto de etiquetas para representar la información morfológica de las palabras. Este conjunto de etiquetas se basa en las etiquetas propuestas por un grupo de investigación y desarrollo lingüístico internacional llamado EAGLES para la anotación morfosintáctica de lexicones y corpus para todas las lenguas europeas. Así pues está previsto que recojan los accidentes gramaticales existentes en las lenguas europeas.
Es por esto que dependiendo de la lengua hay atributos que pueden no especificarse en español. Si un atributo no se especifica significa que, o bien expresa un tipo de información que no existe en la lengua, o que la información no se considera relevante. La infra-especificación de un atributo se marca con el 0 (número cero).
A continuación presentamos las etiquetas que el analizador morfológico utiliza para el castellano en formato de tabla y algunos ejemplos de cada categoría.
Para cada categoría se presentan los atributos, valores y códigos que puede tomar:

ETIQUETAS			
Posición	**Atributo**	**Valor**	**Código**
Columna 1	*Columna 2*	*Columna 3*	*Columna 4*

En la *columna 1* encontramos un número que hace referencia al orden y posición en que aparecen los atributos. La *columna 2* hace referencia a los atributos, el número de los cuales varía dependiendo de la categoría. En la *columna 3* encontramos los valores que puede tomar cada atributo y, finalmente, la *columna 4* representa los códigos que se han establecido para su representación. Las etiquetas en sí sólo son los códigos (columna 4) y se sabe a qué atributo pertenecen por la posición (columna 1) en la que se encuentran.
ADJETIVOS
ADVERBIOS
DETERMINANTES
NOMBRES
VERBOS
PRONOMBRES
CONJUNCIONES
INTERJECCIONES
PREPOSICIONES
SIGNOS DE PUNTUACIÓN
CIFRAS
FECHAS y HORAS
INTERNET y REDES SOCIALES: Uri, Url, em@ils, Hashtags y Twitter-Names

1. ADJETIVOS

ADJETIVOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Adjetivo	A
2	Tipo	Calificativo	Q
		Ordinal	O
		Posesivo	P
		Cromático	K
		Indefinido	I
		Cuantificador	C
		-	0
3	Grado	-	0

		Apreciativo	A
4	Género	Masculino	M
		Femenino	F
		Común	C
5	Número	Singular	S
		Plural	P
		Invariable	N
6	Función	-	0
		Participio	P

1.1. Adjetivos calificativos

- El lema de los adjetivos calificativos será siempre la forma masculina singular (*bonito*) o la forma singular si el adjetivo es de género común (*alegre*). Para los adjetivos invariables, es decir, aquellos que tanto para el singular como para el plural presentan la misma forma, el lema y la forma han de coincidir.
- El valor del último dígito será normalmente **0**. Tan sólo aquellos adjetivos que tengan función de participio tendrán una **P**.
- El atributo *grado* (aún por implementar) se especificará para aquellos adjetivos que tengan grado (comparativos, superlativos) y sufijación apreciativa (aumentativos, despectivos, etc.). Se distinguirán porque el tercer dígito de la etiqueta será **A** mientras que para el resto de adjetivos este valor será **0**.

Ejemplos:

Forma	Lema	Etiqueta
alegres	alegre	AQ0CP0
alegre	alegre	AQ0CS0
bonitas	bonito	AQ0FP0
bonita	bonito	AQ0FS0
bonitos	bonito	AQ0MP0
bonito	bonito	AQ0MS0
antiarrugas	antiarrugas	AQ0CN0
quemada	quemado	AQ0FSP

1.2. Adjetivos Ordinales

- Los adjetivos de tipo ordinal tendrán como lema el masculino singular siendo también la forma masculina singular plena (*primero*, *tercero*) el lema de las formas apocopadas (*primer*, *tercer*).

Ejemplos:

Forma	Lema	Etiqueta
primer	primero	AO0MS0
primera	primero	AO0FS0
primeras	primero	AO0FP0
primero	primero	AO0MS0
primeros	primero	AO0MP0

2. ADVERBIOS

ADVERBIOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Adverbio	R
2	Tipo	General	G
		Negativo	N
		Afirmativo	A
		Modo	M
		Temporal	T
		Lugar	L
		Cantidad	Q
		Duda	D
		Deseo	W
		Comparativo	C
		Frecuencia	F
		Correlativo	K
		Exclamativo	E
		Cuantificador	H
		Interrogativo	I
		Relativo	R
		Familiar	F
		Meta	Z
		Exclamativo Pronominal	X

- Para los adverbios, de momento, tan sólo indicamos que es de tipo general o de tipo negativo.
- La etiqueta de adverbio negativo (RN) está reservada exclusivamente para el adverbio *no*.
- Esta etiqueta sirve tanto para los adverbios y para las locuciones adverbiales.
- El lema de los adverbios acabados en -mente es la misma forma adverbial acabada en -mente, es decir, el lema de *rápidamente* será *rápidamente*.

Ejemplos:

Forma	Lema	Etiqueta
despacio	=	RM
hábilmente, sólidamente, xxx_mente	=	RM
a_cuatro_patas	=	RM
Ahora	=	RT
Siempre	=	RT
talvez, acaso, quizá, igual	=	RD
Antes, después, posteriormente	=	RO
cuan	=	RK

más, menos	=	RC
a_granel	=	RQ
no	=	RN
si	=	RA
donde, adonde	=	RI
diariamente, semanalmente, anualmente	=	RF
muy, mas, tan, apenas, menos	=	RH
adonde, cual, cuanto cuanta, donde	=	RR
también, además	=	RZ
que	=	RX
ojalá	=	RW

3. DETERMINANTES

DETERMINANTES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Determinante	D
2	Tipo	Demostrativo	D
		Posesivo	P
		Interrogativo	T
		Exclamativo	E
		Indefinido	I
		Artículo	A
		Cardinal	C
		Numeral	N
3	Persona	Primera	1
		Segunda	2
		Tercera	3
4	Género	Masculino	M
		Femenino	F
		Común	C
		Neutro	N
5	Número	Singular	S
		Plural	P
		Invariable	N
6	Poseedor	Singular	S

		Plural	P

- El atributo *Persona* tendrá por defecto el valor 0, con excepción de los determinantes posesivos que podrán tomar el valor 1, 2 y 3.
- El atributo *Poseedor* sólo se especificará para los determinantes posesivos. Si el referente es singular, el valor será *S*, si es plural el valor será *P*. Cuando el referente sea de tercera persona el valor de este atributo será 0 ya que es imposible distinguir los referentes *de él-de ellos*.

Ejemplos:

3.1. Determinantes Demostrativos

Forma	Lema	Etiqueta
aquel	aquel	DD0MS0
aquella	aquel	DD0FS0
aquellas	aquel	DD0FP0
aquellos	aquel	DD0MP0
esa	ese	DD0FS0
esas	ese	DD0FP0
ese	ese	DD0MS0
esos	ese	DD0MP0
esta	este	DD0FS0
estas	este	DD0FP0
este	este	DD0MS0
estos	este	DD0MP0
tal	tal	DD0CS0
tales	tal	DD0CP0
semejante	semejante	DD0CS0
semejantes	semejante	DD0CP0

3.2. Determinantes Posesivos

Forma	Lema	Etiqueta
mi	mi	DP1CSS
mis	mi	DP1CPS
mía	mío	DP1FSS
mías	mío	DP1FPS
mío	mío	DP1MSS
míos	mío	DP1MPS
tu	tu	DP2CSS
tus	tu	DP2CPS
tuya	tuyo	DP2FSS
tuyas	tuyo	DP2FPS
tuyo	tuyo	DP2MSS
tuyos	tuyo	DP2MPS
su	su	DP3CS0
sus	su	DP3CP0
nuestra	nuestro	DP1FSP
nuestras	nuestro	DP1FPP
nuestro	nuestro	DP1MSP

nuestros	nuestro	DP1MPP
vuestra	vuestro	DP2FSP
vuestras	vuestro	DP2FPP
vuestro	vuestro	DP2MSP
vuestros	vuestro	DP2MPP
suya	suyo	DP3FS0
suyas	suyo	DP3FP0
suyo	suyo	DP3MS0
suyos	suyo	DP3MP0

3.3. Determinantes Interrogativos

Forma	Lema	Etiqueta
cuánta	cuánto	DT0FS0
cuántas	cuánto	DT0FP0
cuánto	cuánto	DT0MS0
cuántos	cuánto	DT0MP0
qué	qué	DT0CN0

3.4. Determinantes Exclamativos

Forma	Lema	Etiqueta
qué	qué	DE0CN0

3.5. Determinantes Indefinidos

Forma	Lema	Etiqueta
alguna	alguno	DI0FS0
algunas	alguno	DI0FP0
alguno	alguno	DI0MS0
algún	alguno	DI0MS0
algunos	alguno	DI0MP0
bastante	bastante	DI0CS0
bastantes	bastante	DI0CP0
cada	cada	DI0CS0
ninguna	ninguno	DI0FS0
ningunas	ninguno	DI0FP0
ninguno	ninguno	DI0MS0
ningún	ninguno	DI0MS0
ningunos	ninguno	DI0MP0
otra	otro	DI0FS0
otras	otro	DI0FP0
otro	otro	DI0MS0
otros	otro	DI0MP0
sendas	sendos	DI0FP0
sendos	sendos	DI0MP0
tantas	tanto	DI0FP0
tanta	tanto	DI0FS0
tantos	tanto	DI0MP0
tanto	tanto	DI0MS0
todas	todo	DI0FP0

toda	todo	DI0FS0
todos	todo	DI0MP0
todo	todo	DI0MS0
unas	uno	DI0FP0
una	uno	DI0FS0
unos	uno	DI0MP0
un	uno	DI0MS0
varias	varios	DI0FP0
varios	varios	DI0MP0

3.6. Artículos

- Sólo tratamos como artículo las formas del *artículo definido*. No contemplamos la categoría de *artículo indefinido* (*un*) puesto que hemos decidido tratarlas como determinantes indefinidos (véase 3.5) o numerales (véase 3.7).

Ejemplos:

Forma	Lema	Etiqueta
el	el	DA0MS0
los	el	DA0MP0
lo	el	DA0NS0
la	el	DA0FS0
las	el	DA0FP0

3.7. Determinantes Numerales

- El lema de los numerales que tengan género y número será la forma masculina singular.
- Dentro de los determinantes numerales cardinales contemplamos los numerales partitivos (*onceavo*, *doceavo*...) y los numerales multiplicativos (*doble*, *triple*...).

Forma	Lema	Etiqueta
veintisiete	veintisiete	DN0CP0
veintiuna	veintiuno	DN0FP0
veintiuno	veintiuno	DN0MP0
veintiún	veintiuno	DN0CP0
doceava	doceavo	DN0FS0
doceavas	doceavo	DN0FP0
doceavo	doceavo	DN0MS0
doceavos	doceavo	DN0MP0
doble	doble	DN0CS0
quíntuple	quíntuple	DN0CS0
quíntuples	quíntuple	DN0CP0

4. NOMBRES

NOMBRES

Pos.	Atributo	Valor	Código
1	Categoría	Nombre	N
2	Tipo	Común	C
		Propio	P
		Entidad	E
		Twitter	T
		Hashtag	H
3	Género	Masculino	M
		Femenino	F
		Común	C
		Ambiguo	A
		Epíceno	E
		Neutro	N
4	Número	Singular	S
		Plural	P
		Invariable	I
		Neutral	N
		Given-Name[63]	G
		Error	?
5	Clasificación Semántica 1	-	0
		Animal	A
	Body	Parte del cuerpo	B
		Ciencia	C
		Dirección	D
		Electronica	E
		oFicio	F
		Geográfico	G
	pHenomen	Fenómeno	H
		Informática	I
	Clasificación Semántica 1		
		Compuesto	J
	Noun act	acto	K
	Native	oriundo	L
	Method	Método	M
		Nominativo	N
	Named Organization	Organización	O
	People / Person	Persona / Humano	P
	Quantity	Cantidad	Q

[63] Este espacio de números, se utiliza para indicar que el sustantivo propio es del tipo de persona única o nombre de persona, resultó de utilidad colocarlo en esta posición para mantener compatibilidad.

	Root (ontologic)	Raíz	R
	Being	Ser	S
	Noun sTate	Estado	T
	Unit	Unidad	U
		Vegetal	V
	Process / Working..	Proceso	W
	Foreign	Origen eXtranjero	X
	Yield	Resultado	Y
	Appliance/Artifact	Aparato	A
		Biological	B
	Food	Comida/Alimento	C
		Doctrina	D
		Entidad	E
	Feeling	Sentimiento	F
		Grupo	G
		Herramienta	H
		Instrumento	I
	Knowledge	Conocimiento	K
		Locación	L
		Motivo	M
		Comunicación	N
		Objecto	O
		Posesión	P
		Relación	R
		Substancia	S
		Tiempo	T
		Estudio	U
		Evento	V
		Medida	W
		Texto	X
		Propiedad	Y
	Shape	Forma	Z
6	Clasificación Semántica 2	-	0
		Artificial	A
		Abreviación / Acrónimo	B
		Computación	C
		Distancia	D
		Efecto	E
	Fish	Pescado	F
		Geographic	G
		Virtual	H

	bIrds	Aves	I
		Reptiles	R
		Anfibios	J
		Mamíferos	K
		Locación	L
		Mineral	M
		Natural	N
		Organización	O
		Sonido	S
		Tiempo	T
	Unknown	Desconocido	U
		Valor	V
		Peso	W
		tóXico	X
	Yield	resultado	Y
		luZ	Z
		Absoluto / Arte	A
		Biología	B
		Cetáceo	C
		Derecho	D
		Enfermedad	E
		Farmacéutico	F
		Figurativo	G
		Física	H
		Ingeniería	I
		Invertebrado	J
		Molusco	K
		Política	L
		Medicina	M
		Anatomía	N
		Crustáceo	O
		Poesía	P
		Bioquímica	Q
		Relativo	R
		Artrópodo	S
		Técnica	T
		Microscópico	U
		Botánica	V
		Relacionado al Agua	W
		Abstracto	X
		Taxón	Y
		Zoología	Z

7	Grado	Apreciativo	A
		Aumentativo	A
		Superlativo	S
	(LexEsp)	Positivo	P
	(LexEsp)	Intensivo	I
		Peyorativo	Y
		Diminutivo	D
	(Eagles 2.0)	Diminutivo	C
		Comparativo	M
		Despectivo	E
		Adversativo	V
		Distributivo	T
		Irónico	O
		Concesivo	N

- Los nombres tienen como lema la forma singular, tanto si es de género femenino como masculino o neutro. Para los nombres invariables, es decir, aquellos que tanto para el singular como para el plural presentan la misma forma (*tesis*), el lema y la forma coincidirán.
- De momento no utilizamos la clasificación semántica. Los nombres propios tendrán la etiqueta NP00000 y NPG cuando sean un nombre de persona (*given-name*).
- El atributo *grado* (aún por implementar) se especificará para aquellos nombres que tengan sufijación apreciativa (aumentativos, despectivos, etc.) y se distinguirán porque el valor de este atributo será **A**. Para el resto de nombres este valor será **0**.

Ejemplos:

Forma	Lema	Etiqueta
chico	chico	NCMS000
chicos	chico	NCMP000
chica	chica	NCFS000
chicas	chica	NCFP000
oyente	oyente	NCCS000
oyentes	oyente	NCCP000
cortapapeles	cortapapeles	NCMN000
tesis	tesis	NCFN000
Barcelona	barcelona	NP00000
COI	coi	NP00000

5. VERBOS

VERBOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Verbo	V
2	Tipo	Principal[64]	M

[64] En el caso de que haya información sobre la funcionalidad del verbo, se especifican las etiquetas VT/VP/VI/VW/VA/VS en caso de no haber información de tipo, se utiliza esta primera: VM.

		Intransitivo	I
		Transitivo	T
		Pronominal	P
		Meteorológico	W
		Defectivo	D
		Auxiliar	A
		Semiauxiliar	S
3	Modo	Indicativo	I
		Subjuntivo	S
		Imperativo	M
		Infinitivo	N
		Gerundio	G
		Participio	P
4	Tiempo	Presente	P
		Imperfecto	I
		Futuro	F
		Pasado	S
		Condicional	C
		-	0
5	Persona	Primera	1
		Segunda	2
		Tercera	3
6	Número	Singular	S
		Plural	P
7	Género	Masculino	M
		Femenino	F

- El lema del verbo ha de ser siempre el infinitivo.
- Etiquetamos las formas del verbo *haber* como auxiliares (VA), las del verbo *ser* como semiauxiliares (VS) y las restantes como principales (VM), salvo que se tenga información del comportamiento del mismo, como ser: si el verbo es Defectivo (VD), Meteorológico, (VW) Transitivo (VT), Intransitivo (VI) y/o Pronominal o Reflexivo (VP), en definitiva, sólo se usa VM es para cuando no se sabe mucho del verbo.
- El atributo de *Género* sólo afecta a los participios, para el resto de formas este atributo no se especifica (0).
- Para las formas de infinitivo y gerundio no se especifican los atributos de *Tiempo*, *Persona*, *Número* y *Género*, por lo que su valor será 0.

Forma	Lema	Etiqueta
cantada	Cantar	VMP00S***F***
cantadas	Cantar	VMP00P***F***
cantado	Cantar	VMP00S***M***
cantados	Cantar	VMP00P***M***

Ejemplos:

Tiempo	VERBOS PRINCIPALES			VERBO SEMIAUXILIAR		
	Forma	Lema	Etiqueta	Forma	Lema	Etiqueta
PRESENTE DE INDICATIVO	canto	cantar	VMIP1S0	soy	ser	VSIP1S0
	cantas	cantar	VMIP2S0	eres	ser	VSIP2S0
	canta	cantar	VMIP3S0	es	ser	VSIP3S0
	cantamos	cantar	VMIP1P0	somos	ser	VSIP1P0
	cantáis	cantar	VMIP2P0	sois	ser	VSIP2P0
	cantan	cantar	VMIP3P0	son	ser	VSIP3P0
PRETÉRITO IMPERFECTO	cantaba	cantar	VMII1S0	era	ser	VSII1S0
	cantabas	cantar	VMII2S0	eras	ser	VSII2S0
	cantaba	cantar	VMII3S0	era	ser	VSII3S0
	cantábamos	cantar	VMII1P0	éramos	ser	VSII1P0
	cantabais	cantar	VMII2P0	erais	ser	VSII2P0
	cantaban	cantar	VMII3P0	eran	ser	VSII3P0
PRETÉRITO PERFECTO SIMPLE	canté	cantar	VMIS1S0	fui	ser	VSIS1S0
	cantaste	cantar	VMIS2S0	fuiste	ser	VSIS2S0
	cantó	cantar	VMIS3S0	fue	ser	VSIS3S0
	cantamos	cantar	VMIS1P0	fuimos	ser	VSIS1P0
	cantasteis	cantar	VMIS2P0	fuisteis	ser	VSIS2P0
	cantaron	cantar	VMIS3P0	fueron	ser	VSIS3P0
FUTURO DE INDICATIVO	cantaré	cantar	VMIF1S0	seré	ser	VSIF1S0
	cantarás	cantar	VMIF2S0	serás	ser	VSIF2S0
	cantará	cantar	VMIF3S0	será	ser	VSIF3S0
	cantaremos	cantar	VMIF1P0	seremos	ser	VSIF1P0
	cantaréis	cantar	VMIF2P0	seréis	ser	VSIF2P0
	cantarán	cantar	VMIF3P0	serán	ser	VSIF3P0
CONDICIONAL	cantaría	cantar	VMIC1S0	sería	ser	VSIC1S0
	cantarías	cantar	VMIC2S0	serías	ser	VSIC2S0
	cantaría	cantar	VMIC3S0	sería	ser	VSIC3S0
	cantaríamos	cantar	VMIC1P0	seríamos	ser	VSIC1P0
	cantaríais	cantar	VMIC2P0	seríais	ser	VSIC2P0
	cantarían	cantar	VMIC3P0	serían	ser	VSIC3P0
PRESENTE DE SUBJUNTIVO	cante	cantar	VMSP1S0	sea	ser	VSSP1S0
	cantes	cantar	VMSP2S0	seas	ser	VSSP2S0
	cante	cantar	VMSP3S0	sea	ser	VSSP3S0
	cantemos	cantar	VMSP1P0	seamos	ser	VSSP1P0
	cantéis	cantar	VMSP2P0	seáis	ser	VSSP2P0
	canten	cantar	VMSP3P0	sean	ser	VSSP3P0
PRETÉRITO IMPERFECTO	cantara	cantar	VMSI1S0	fuera	ser	VSSI1S0
	cantaras	cantar	VMSI2S0	fueras	ser	VSSI2S0
	cantara	cantar	VMSI3S0	fuera	ser	VSSI3S0

	cantáramos	cantar	VMSI1P0	fuéramos	ser	VSSI1P0
	cantarais	cantar	VMSI2P0	fuerais	ser	VSSI2P0
	cantaran	cantar	VMSI3P0	fueran	ser	VSSI3P0
	cantase	cantar	VMSI1S0	fuese	ser	VSSI1S0
	cantases	cantar	VMSI2S0	fueses	ser	VSSI2S0
	cantase	cantar	VMSI3S0	fuese	ser	VSSI3S0
	cantásemos	cantar	VMSI1P0	fuésemos	ser	VSSI1P0
	cantaseis	cantar	VMSI2P0	fueseis	ser	VSSI2P0
	cantasen	cantar	VMSI3P0	fuesen	ser	VSSI3P0
FUTURO DE SUBJUNTIVO	cantare	cantar	VMSF1S0	fuere	ser	VSSF1S0
	cantares	cantar	VMSF2S0	fueres	ser	VSSF2S0
	cantare	cantar	VMSF3S0	fuere	ser	VSSF3S0
	cantáremos	cantar	VMSF1P0	fuéremos	ser	VSSF1P0
	cantareis	cantar	VMSF2P0	fuereis	ser	VSSF2P0
	cantaren	cantar	VMSF3P0	fueren	ser	VSSF3P0
GERUNDIO	cantando	cantar	VMG0000	siendo	ser	VSG0000
IMPERATIVO	canta	cantar	VMM02S0	sé	ser	VSM02S0
	cante	cantar	VMM03S0	sea	ser	VSM03S0
	cantemos	cantar	VMM01P0	seamos	ser	VSM01P0
	cantad	cantar	VMM02P0	sed	ser	VSM02P0
	canten	cantar	VMM03P0	sean	ser	VSM03P0
INFINITIVO	cantar	cantar	VMN0000	ser	ser	VSN0000
PARTICIPIO	cantada	cantar	VMP00SF	sido	ser	VSP00SM
	cantado	cantar	VMP00SM			
	cantadas	cantar	VMP00PF			
	cantados	cantar	VMP00PM			

6. PRONOMBRES

PRONOMBRES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Pronombre	P
2	Tipo	Personal	P
		Demostrativo	D
		Se	S
		Posesivo	X
		Indefinido	I
		Interrogativo	T
		Exclamativo	E
		Clítico	C
		Átono	A
		Relativo	R

		Numeral	N
		Exclamativo	E
3	Persona	Primera	1
		Segunda	2
		Tercera	3
4	Género	Masculino	M
		Femenino	F
		Común	C
		Neutro	N
5	Número	Singular	S
		Plural	P
		Invariable	N
6	Caso	Nominativo	N
		Acusativo	A
		Dativo	D
		Oblicuo	O
7	Poseedor	Singular	S
		Plural	P
8	Politeness	Polite	P

- El atributo *Persona* se especificará para los pronombres personales y posesivos, para el resto de formas el valor será 0.
- El atributo *Caso* es específico para los pronombres personales, para el resto será 0.
- El atributo *Poseedor* sólo se usará con los pronombres posesivos para marcar el número del poseedor: singular para *el_mío*, *el_tuyo*, plural para *el_nuestro* y *el_vuestro*. Para los pronombres en que el poseedor es una tercera persona (*el_suyo*), como no se podrá distinguir si es de singular o plural (si se refiere *a él* o *a ellos*), este atributo tomará valor 0.
- El atributo *Politeness* (cortesía) se especificará para los pronombres personales *usted*, *ustedes* y *vos*.
- El lema será siempre la forma masculina singular. Por ejemplo, en el caso de los pronombres personales los lemas serán *yo*, *tú y él*.

6.1. Pronombres Personales

Forma	Lema	Etiqueta
yo	Yo	PP1CSN00
me	yo	PP1CS000
mí	yo	PP1CSO00
nos	yo	PP1CP000
nosotras	yo	PP1FP000
nosotros	yo	PP1MP000
conmigo	yo	PP1CSO00
te	tú	PP2CS000
ti	tu	PP2CSO00
tú	tú	PP2CSN00
os	tú	PP2CP000
usted	tú	PP2CS00P

ustedes	tú	PP2CP00P
vos	tú	PP3CN00P
vosotras	tú	PP2FP000
vosotros	tú	PP2MP000
contigo	tú	PP2CNO00
él	él	PP3MS000
ella	él	PP3FS000
ellas	él	PP3FP000
ello	él	PP3NS000
ellos	él	PP3MP000
la	él	PP3FSA00
las	él	PP3FPA00
lo	él	PP3MSA00
lo	él	PP3CNA00
los	él	PP3MPA00
le	él	PP3CSD00
les	él	PP3CPD00
se	él	PP3CN000
sí	él	PP3CNO00
consigo	él	PP3CNO00

6.2. Pronombres Demostrativos

Forma	Lema	Etiqueta
aquéllas	aquel	PD0FP000
aquélla	aquel	PD0FS000
aquéllos	aquel	PD0MP000
aquél	aquel	PD0MS000
aquellas	aquel	PD0FP000
aquella	aquel	PD0FS000
aquellos	aquel	PD0MP000
aquel	aquel	PD0MS000
aquello	aquel	PD0NS000
ésas	ese	PD0FP000
ésa	ese	PD0FS000
esas	ese	PD0FP000
esa	ese	PD0FS000
esos	ese	PD0MP000
ese	ese	PD0MS000
ésos	ese	PD0MP000
ése	ese	PD0MS000
eso	ese	PD0NS000
esotra	esotro	PD0FS000
esotro	esotro	PD0MS000
esta	este	PD0FS000
éstas	este	PD0FP000
ésta	este	PD0FS000
estas	este	PD0FP000

esta	este	PD0FS000
estos	este	PD0MP000
este	este	PD0MS000
éstos	este	PD0MP000
éste	este	PD0MS000
esto	este	PD0NS000
estotra	estotro	PD0FS000
estotro	estotro	PD0MS000
tal	tal	PD0CS000
tales	tal	PD0CP000

6.3. Pronombres Posesivos

Forma	Lema	Etiqueta
la_mía	el_mío	PX1FS0S0
las_mías	el_mío	PX1FP0S0
el_mío	el_mío	PX1MS0S0
los_míos	el_mío	PX1MP0S0
lo_mío	el_mío	PX1NS0S0
la_nuestra	el_nuestro	PX1FS0P0
las_nuestras	el_nuestro	PX1FP0P0
el_nuestro	el_nuestro	PX1MS0P0
los_nuestros	el_nuestro	PX1MP0P0
lo_nuestro	el_nuestro	PX1NS0P0
la_suya	el_suyo	PX3FS000
las_suyas	el_suyo	PX3FP000
el_suyo	el_suyo	PX3MS000
los_suyos	el_suyo	PX3MP000
lo_suyo	el_suyo	PX3NS000
la_tuya	el_tuyo	PX2FS0S0
las_tuyas	el_tuyo	PX2FP0S0
el_tuyo	el_tuyo	PX2MS0S0
los_tuyos	el_tuyo	PX2MP0S0
lo_tuyo	el_tuyo	PX2NS0S0
la_vuestra	el_vuestro	PX2FS0P0
las_vuestras	el_vuestro	PX2FP0P0
el_vuestro	el_vuestro	PX2MS0P0
los_vuestros	el_vuestro	PX2MP0P0
lo_vuestro	el_vuestro	PX2NS0P0

6.4. Pronombres Indefinidos

Forma	Lema	Etiqueta
algo	algo	PI0CS000
alguien	alguien	PI0CS000
alguna	alguno	PI0FS000
algunas	alguno	PI0FP000
alguno	alguno	PI0MS000
algunos	alguno	PI0MP000

bastante	bastante	PI0MS000
bastantes	bastante	PI0MP000
cualesquiera	cualquiera	PI0CP000
cualquiera	cualquiera	PI0CS000
demás	demás	PI0CP000
misma	mismo	PI0FS000
mismas	mismo	PI0FP000
mismo	mismo	PI0MS000
mismos	mismo	PI0MP000
mucha	mucho	PI0FS000
muchas	mucho	PI0FP000
mucho	mucho	PI0MS000
muchos	mucho	PI0MP000
nada	nada	PI0CS000
nadie	nadie	PI0CS000
ninguna	ninguno	PI0FS000
ningunas	ninguno	PI0FP000
ninguno	ninguno	PI0MS000
ningunos	ninguno	PI0MP000
otra	otro	PI0FS000
otras	otro	PI0FP000
otro	otro	PI0MS000
otros	otro	PI0MP000
poca	poco	PI0FS000
pocas	poco	PI0FP000
poco	poco	PI0MS000
pocos	poco	PI0MP000
quienquier	quienquiera	PI0CS000
quienesquiera	quienquiera	PI0CP000
quienquiera	quienquiera	PI0CS000
tanta	tanto	PI0FS000
tantas	tanto	PI0FP000
tanto	tanto	PI0MS000
tantos	tanto	PI0MP000
toda	todo	PI0FS000
todas	todo	PI0FP000
todo	todo	PI0MS000
todos	todo	PI0MP000
una	uno	PI0FS000
unas	uno	PI0FP000
uno	uno	PI0MS000
unos	uno	PI0MP000
varias	varios	PI0FP000
varios	varios	PI0MP000

6.5. Pronombres Interrogativos

Forma	Lema	Etiqueta

adónde	adónde	PT000000
cómo	cómo	PT000000
cuál	cuál	PT0CS000
cuáles	cuál	PT0CP000
cuándo	cuándo	PT000000
cuánta	cuánto	PT0FS000
cuántas	cuánto	PT0FP000
cuánto	cuánto	PT0MS000
cuántos	cuánto	PT0MP000
dónde	dónde	PT000000
qué	qué	PT0CS000
quién	quién	PT0CS000
quiénes	quién	PT0CP000

6.6. Pronombres Relativos

Forma	Lema	Etiqueta
como	como	PR000000
donde	donde	PR000000
adonde	adonde	PR000000
cuando	cuando	PR000000
cual	cual	PR0CS000
cuales	cual	PR0CP000
cuanta	cuanto	PR0FS000
cuantas	cuanto	PR0FP000
cuanto	cuanto	PR0MS000
cuantos	cuanto	PR0MP000
cuya	cuyo	PR0FS000
cuyas	cuyo	PR0FP000
cuyo	cuyo	PR0MS000
cuyos	cuyo	PR0MP000
que	que	PR0CN000
quien	quien	PR0CS000
quienes	quien	PR0CP000

6.7. Pronombres Numerales

Forma	Lema	Etiqueta
catorce	catorce	PN0CP000
cien	cien	PN0CP000
cinco	cinco	PN0CP000
cincuenta	cincuenta	PN0CP000
cuatro	cuatro	PN0CP000
cuatrocientas	cuatrocientos	PN0FP000
cuatrocientos	cuatrocientos	PN0MP000
diez	diez	PN0CP000
doce	doce	PN0CP000
dos	dos	PN0CP000

una	uno	PN0FS000
uno	uno	PN0MS000
un	uno	PN0MS000

6.8. Pronombres Exclamativos

Forma	Lema	Etiqueta
qué	qué	PE000000

7. CONJUNCIONES

CONJUNCIONES			
Pos.	**Atributo**	**Valor**	**Código>**
1	Categoría	Conjunción	C
2	Tipo	Coordinada	C
		Subordinada	S
		Adversativa	A
		Casual	K
		Comparativa	M
		Concesiva	N
		Copulativa	P
		Distributiva	D
		Final	F
		Ilativa	I
		Temporal	T
		Condicional	Q
		Disyuntiva	O

7.1. Conjunción Coordinada

Forma	Lema	Etiqueta
e	e	CY
empero	empero	CC
mas	mas	CC
ni	ni	CY
o	o	CO
ora	ora	CC
pero	pero	CC
sino	sino	CC
siquiera	siquiera	CC
u	u	CO
y	y	CY

7.2. Conjunción Subordinada

Forma	Lema	Etiqueta
aunque	aunque	CS

como	como	CS
conque	conque	CS
cuando	cuando	CS
donde	donde	CS
entonces	entonces	CS
ergo	ergo	CS
incluso	incluso	CS
luego	luego	CT
mientras	mientras	CT
porque	porque	CF, CK
pues	pues	CS
que	que	CS
sea	sea	CS
si	si	CS
ya	ya	CS

Notas: Se especifican a continuación cómo se re-clasificaron las conjunciones, en caso de ser muy ambigua, se especifica como coordinada, en otos casos serán las de abajo como ejemplo:

Coordinada	CC : (sin clasificar)
Subordinada	CS : que
Adversativa	CA : ahora cuanto antes
Causal	CK : porque ca car onde
Comparativa	CM : así aunque maguer magera
Concesiva	CN : así aunque maguer magera
Copulativa	CP : y e ni
Distributiva	CD : ahora ora agora siquier
Disyuntiva	CO : o u
Final	CF : porque
Ilativa	CI : conque entonces luego
Temporal	CT : apenas cuando mientras recién
Condicional	CQ : siempre que, debido a, por

8. INTERJECCIONES

INTERJECCIONES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Interjección	I

Ejemplos:

Forma	Lema	Etiqueta
ah	ah	I
eh	eh	I
ejem	ejem	I
ele	ele	I

9. PREPOSICIONES

PREPOSICIONES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Adposición	S

2	Tipo	Preposición	P
3	Forma	Simple	S
		Contraída	C
3	Género	Masculino	M
4	Número	Singular	S

- Los atributos de *género* y *número* tan sólo se especifican para las preposiciones contraídas **al** y **del**.
- Estas etiquetas también se usan para las locuciones preposicionales.

Ejemplos:

Forma	Lema	Etiqueta
al	al	SPCMS
del	del	SPCMS
a	a	SPS00
ante	ante	SPS00
bajo	bajo	SPS00
cabe	cabe	SPS00
con	con	SPS00
a_partir_de	a_partir_de	SPS00
a_causa_del	a_causa_del	SPCMS

10. SIGNOS DE PUNTUACIÓN

SIGNOS DE PUNTUACIÓN			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Puntuación	F

Ejemplos:

Forma	Lema	Etiqueta
¡	¡	Faa
!	!	Fat
,	,	Fc
[	[	Fca
]	]	Fct
:	:	Fd
"	"	Fe
-	-	Fg
/	/	Fh
¿	¿	Fia
?	?	Fit
{	{	Fla
}	}	Flt
.	.	Fp
(	(	Fpa
)	)	Fpt

«	«	Fra
»	»	Frc
...	...	Fs
%	%	Ft
;	;	Fx
_	_	Fz
+	+	Fz
=	=	Fz

11. CIFRAS

CIFRAS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Cifra	Z
2	Tipo	Moneda	M
		Unidad Física	U
		Química	Q
		Proporción / Procentaje	P
		Partitivo	D
		Moneda	M

- Las cifras se etiquetarán con Z. Bajo esta etiqueta encontraremos: direcciones, números de teléfono, tanteos, etc.
- Las cantidades monetarias recibirán la etiqueta Zm, tendrán como lema la cantidad (en cifras) y el nombre de la unidad monetaria en singular.

Ejemplos:

Forma	Lema	Etiqueta
239	239	DN
+1.25e-25		DN
doscientos_veinte	220	DN
H2O	H2O	Zq
10%	10	Zp
dólares	2000_dólar	Zm
metros	m	Zu
voltios	V	Zu
medio	medio	Zd

12. FECHAS Y HORAS

FECHAS Y HORAS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Fecha/Hora	W
2	Tipo	Fecha	f
		Hora	t

		Día + Hora	x
		Periódico	p
3	SubTipo	Día de la semana	j
		Mes del año	M

Ejemplos:

Forma	Lema	Etiqueta
viernes_26_de_septiembre_de_1992	[viernes:26/9/1992:??.??]	Wx
marzo_de_1954	[??:??/03/1954:??.??]	Wf
siglo_XIX	[s:xix]	Wf
año_1987	[??:??/??/1987:??.??]	Wf
cinco_de_la_mañana	[??:??/??/??:05.00]	Wt

13. Abreviaturas

ABREVIATURAS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Abreviatura	Y
2	Tipo	Titulo (persona)	t
		Nobiliario	n
		Monetaria	m
		Unidad Física	u
		Sustantiva	s
		Adjetiva	a
		Geográfica	g
		Pronominal	p
		Entidad	e
		Adverbial	r
		Named_END	E
		Named_Start	S
		-	0

Ejemplos:

Sr. Yt
Ing. Yt.
von Yn
s.r.l. YE
USD Ym
kg. Yu
RDBMS. Ys
NY Yg
B.M. Ye

Notas: Estas abreviaturas, están basadas en las etiquetas que se obtienen de analizar las más de 6500 abreviaturas internacionalmente comunes, incluidas las de la R.A.E. y otras, la clasificación se ha realizado con vistas de un analizador más sofisticado de entidades nombradas, tendiente a reconocer cosas como nombres de empresas o personas, con varios apellidos, títulos de oficio o de nobleza.

14. INTERNET, Mail, Twitter y Facebook

INTERNET / COMUNICACIONES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Unidad	U
2	Clase	Email	E
		Http	H
		Https	S
		Otro-protocolo	O
		Ftp	F
		Url	L
		Uri	R
		ip	I
		Número Telefónico	T

Ejemplos

minombre.miapellido@miservidor.com **UE**

http//blog.micosa.com/mip'gina_2/pepe.html?5 **UH**

+54 11 5461 5896 **UT**

Notas: Estas 'pseudo' palabras son cada vez más frecuentes en diálogos y textos. Debido a la enorme complejidad de sus definiciones (basadas en varias RFC) que no se las pude tratar apropiadamente y por este motivo se las clasifica (etiqueta) para poder ser tomadas como un 'todo' por cualquier sistema lingüístico y al menos saber a que norma o sistema pertenecen, extrayendo un poco de semántica de estas cosas tan 'extrañas' para un lingüista.

15. ESPECIALES

ESPECIALES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Desconocido	K
2	Tipo	Alfabético	c
		Alfanumérico	a
		Palabra	w
		Puntuación	p

Ejemplos

abcdeghjkl **Kc**

a3b4kli89sjh **Ka**

.,/*^%#!aca.e **Kp**

Notas: Estas palabras se las clasifica de este modo cuasi-mecánico (mediante expresiones regulares) y caen en esta clasificación si todo análisis morfológico ha fallado y no se ha podido detectar tampoco un idioma satisfactoriamente y luego de que el analizador de abreviaturas tampoco ha podido interpretar las puntuaciones con cierta regularidad.

15. RESIDUALES

RESIDUALES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Residual	X
2,3	ISO 639 (2 letras) del Lenguaje Estimado	Español	es
		Inglés	en
		Alemán	de
		Portugués	po

Ejemplos

hohenzoller, altenberger: **Xde**

barbariondo, mujicalainez: **Xes**

castauriçón, munorçálvez: **Xpo**

dragonswaresh, worcesterilsh: **Xen**

Notas: Estas palabras se las clasifica de este modo cuando se detectan con una verosimilitud aceptable de que sea de un determinado idioma y "pronunciable" (es una estimación fonética con más del 75% de certeza) Pero al no estar escritas ebn base de datos alguna ni reconocidas como apellidos comunes o nombres de lugares geográficos, íconos u otro nombre propio, son clasificadas con esta etiqueta siempre. Si la palabra está escrita comenzando con mayúsculas, se le estimará (agregará) un nombre propio (NP) como etiqueta alternativa.

16. ERRORES TIPO BASURA

BASURA			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Palabra Errada	B

Ejemplos

kljhnjdyhnsghrioyhedf: **B**

Notas: Estas no-palabras se las clasifica a partir de que no poseen característica alguna de pertenecer a un lenguaje conocido y suelen ser cuando se ingresan datos adrede mal, como se dice "*cuando un gato camina por el teclado*" ergo, se las considera como basura o Bad-Words.

Apéndice 3

Alfabeto SAMPA

Listado del alfabeto SAMPA extendido del español, utilizado en la conversión léxica-fonética (de grafema a fonema sampa), para el algoritmo de métrica de similitud fonética entre formas escritas.

Este listado contiene una inclusión de ciertos fonemas equivalentes a variantes latinoamericanas, en especial las rioplatenses, junto con las ibéricas, para poder hacer un conjunto máximo común.

	IPA	sampa	name	Description
0				Silencio-espacio-nada,muda/?|?
1	i	i	i	Vocal,cerrada-anterior,expresa/B(i)s|Bis
2	e	e	e	Vocal,cerrada-medio-anterior,expresa/M(e)s|Mes
3	a	a	a	Vocal,abierta-central,expresa/M(a)s|Mas
4	o	o	o	Vocal,cerrada-medio-posterior-redon,expresa/T(o)s|Tos
5	u	u	u	Vocal,cerrada-posterior-redon,expresa/Tul|Tul
6	j	j	i	Approximant,palatal,expresa/LaB(i)o|Labio
7	w	w	u	Approximant,velar-labial,expresa/AG(w)a|Agua
8	l	l	l	Approximant,apico-alveolar,expresa/(L)obo|Lobo
9	m	m	m	Nasal,labial,expresa/(M)esa|Mesa
10	M	M	m	Nasal,labio-dental,expresa/E(m)fermo|Enfermo
10	n	n	n	Nasal,apico-alveolar,expresa/(N)ada|Nada
11	N	N	n	Nasal,velar,expresa/O(N)go|Hongo
12	j	J	ñ	Nasal,dorso-palatal,expresa/Ni(J)o|Niño
13	b	B	B	Approximant,labial,expresa/Tu(B)o|Tubo
14	d	D	D	Approximant,apico-dental,expresa/A(d)a|Hada
15	g	G	G	Approximant,velar-dorsal,expresa/A(g)osto|Agosto
16	b	b	b	Oclusiva,labial,expresa/(B)eso|Beso
17	d	d	d	Oclusiva,apico-dental,expresa/(D)ar|Dar
18	g	g	g	Oclusiva,velar-dorsal,expresa/(G)ula|Gula
19	r	r	r	Oclusiva,apico-alveolar,expresa/Pe(r)o|Pero
20	r	R	rr	Trina,apico-alveolar,expresa/Ca(R)o|Carro
21	l	L	Y	Lateral,dorso-palatal,expresa/(L)uBja|Lluvia
21	l	Z	Y	Lateral,dorso-palatal,expresa/(L)uBja|Lluvia
21	l	Y	Y	Lateral,dorso-palatal,expresa/(L)uBja|Lluvia
21	l	y	y	Lateral,dorso-palatal,expresa/(L)uBja|Lluvia
21	z	T	s	Fricativa,interdental,muda/(z)ona|Tona
22	h	h	s	Fricativa,gloatal,muda/A(h)ta|Hasta
23	p	p	p	Oclusiva,labial,muda/(P)ala|Pala
24	t	t	t	Oclusiva,apico-dental,muda/(T)ierra|Tierra
25	k	k	k	Oclusiva,velar-dorsal,muda/(K)ilo|Kilo
26	ts	H	ch	Africada,dorso-palatal,muda/Te(H)o|Techo
27	s	s	s	Fricativa,dorso-alveolar,muda/(S)ala|Sala
27	ç	S	s	Fricativa,dorso-alveolar,muda/(S)ala|Sala
28	f	f	f	Fricativa,labio-dental,muda/(F)e|Fe

29	x	x	J	Fricativa,velar-dorsal,muda/(X)wez|Juez
30	?	C	j	Fricativa,dorso-palatal,muda/Ar(C)entina|Argentina
31	W	W	wh	Vocal,cerrada-posterior-redon,expresa (ingles) What
32	I	I	y	Vocal,semi-cerrada-anterior,expresa (ingles) (my) (boy)
33	u	U	u	Vocal,semi-cerrada-posterior-redon,expresa (ingles) how: haU
34	a	A	a	Vocal,abierta-central,expresa (ingles)
35	ks	X	ks	Oclusiva-Fricativa,velar-dorsal,muda/se(x)o

Apéndice 4

Fórmulas Químicas

Esta sección no pretende ser un tratado de química, ni mucho menos; la intención es solo dar un pantallazo de las nomenclaturas, pero si vale como una introducción a algunas de las extraordinarias capacidades que posee el **Reconocedor de Entidades Nombradas (NER)**, en este caso aplicado a ***Fórmulas Químicas***, en especial tratándose de reconocer apropiadamente los ingresos de datos del tipo de fórmulas químicas sencillas *(hasta bastante complejas)* y también reconocerlas por sus nombres asociados.

Detalle de Reglas de Reconocimiento

Explicación del módulo de reconocimiento de secuencias de texto que representan fórmulas químicas simples, realizando un análisis pormenorizado pero básico.

Ejemplos de Texto denotando entidades químicas:

> **H2O** *(simple palabra, formato IUPAC)*
> **CO3Ca+nH2O** *(múltiples palabras, formato IUPAC)*
> **ácido bezóico** *(término químico usando nombres estándar)*

Ahora como existen muchas convenciones y formas de expresar lo mismo mediante permutaciones y agrupaciones de los mismos átomos y partes, pues también se podría colocar en forma menos elegante sin faltar a la verdad: OH2, (H)2(O), H2(O), OH2, O(H)2, entre otras variantes de forma y posición, resultando equivalentes y correspondiendo a la misma molécula, solamente reconocible para un profesional químico avezado.

Debido a todo esto y al hecho de que hay una gran variabilidad en la forma y convenciones para escribir los compuestos químicos, es que hemos rescatado un denominador común con las siguientes limitaciones y reglas, reconociendo por ejemplo:

```
H2O          agua

NO3-NH4      nitrato de amonio
```

Se reconoce notación tipo IUPAC, simplificada e incluyendo la expresión de los enlaces químicos con signos '-' y '=', se permiten paréntesis de agrupación, también números detrás, indicando el número de átomos o grupos funcionales y delante, indicando en número total, incluido 'n' como genérico, ej.:

```
(MgO)3-(SiO2)4-H2O            talco (silicato de magnesio)

Ca(OH)2+nH2O      agua blanca (hidróxido de calcio en solución)
```

También se reconocen los nombres de algunos productos usuales del mercado, por ejemplo:

```
ácido bórico, permanganato de potasio, hidróxido de zinc, etc.
```

El sistema actualmente reconoce la fórmula química en forma estequiométrica, es decir prescinde de la estructura dado a que este sistema no pretende ser un tratado de química ni nada parecido, tan solo entender nativamente "un poco" de que se trata esto de la química.

Lo importante, es el poder reconocer una estructura química simple posiblemente escrita por un usuario no-experto y como cosa deseable el 'saber que clase de cosa se trata' o en su defecto el "nombre-comercial" del producto que está expresando por esa fórmula.

Por el funcionamiento avanzado de sus algoritmos, el clasificador halla naturalmente y sin esfuerzos todos los isómeros químicos de una fórmula reconocida, (*productos con igual número y tipo de átomos*).

Luego el clasificador los representa internamente como alternativa válida (entidad química) abarcando todas las posiciones de la frase de entrada en donde se agruparon las palabras y signos ingresados por un usuario. Esto es de suma utilidad ya que es muy complejo hallar los límites entre una gramática química y la gramática del lenguaje mismo, en una misma frase.

Reglas de Escritura Química Reconocidas

Mencionaremos la lista de algunas de las reglas que aceptamos en el análisis, a continuación:

Regla 1:

Los nombres de los 121 elementos (átomos), deben estar escritos correctamente, con las mayúsculas y minúsculas tal como se los nombra en la tabla periódica. Por ejemplo: 'Ag' es reconocido como el símbolo químico de la Plata *(viene del Argirium - en latín),* y no 'AG' , 'ag' ni 'aG' esto además es independiente del idioma en cuestión.

Regla 2:

Se permite usar paréntesis para agrupar bloques funcionales dentro de las moléculas, como ser:

NO3(OH)2

Regla 3:

Los números pueden estar delante o detrás, significando cosas diferentes dependiendo del caso, por ejemplo: **(NO3)2** es aproximadamente equivalente a N2O6 lo es el conteo de átomos, pero no así en su formulación. Esto da lugar a 'similitudes' llamadas isomerías *(isómeros por el número, tipo y cantidad de átomos, y no así por su forma de unión).*

Regla 4:

Se permiten estas isomerías y el sistema considera la 'raíz' de una palabra química el isómero de cantidad, tipo y número de componentes, reconociendo como 'sinónimos aproximados' fórmulas químicas que pueden ser compuestos diferentes pero han sido formulados en forma diferente.

Como veníamos explicando en la regla 3, la posición de los números puede ser delante, o sea (NO3)2 es exactamente equivalente a 2(NO3) y también a 2NO3, solo con una diferente convención de escritura, la cual reconocemos como idéntica.

Regla 5:

En caso de colocar fórmulas de este tipo: C6H2N4 esta fórmula es equivalente a 2(C3HN2) y si bien el autor o químico puede haber intentado plasmar algo de la estructura en la fórmula, la mayoría de las veces esto no es así y se trata de isómeros, por lo cual el analizador los toma como equivalentes, realizando la tarea de factoreo, para hallar equivalentes en su base de conocimientos químicos.

Regla 6:

El conocimiento químico del Analizador y Reconocedor de Entidades Químicas incluye las fórmulas de aproximadamente los 500 elementos y compuestos más comunes, tanto orgánicos como inorgánicos, siendo su capacidad de reconocimiento equivalente al un estudiante universitario medio, habiendo cursado las químicas tanto inorgánicas como orgánica básicas.

Regla 7:

Muchas veces se escribe con uniones mediante guiones: NO2-OH para representar grupos de átomos, esto es reconocido por el sistema, con las siguientes convenciones:

Tipo de unión:

"":	unspecified
"-":	covalent or unspecified
"=":	hidrogen_bridge
"~":	van_der_waals
"·":	hidrogen_bridge
"#":	aleation (no chemical union)

Si bien se puede especificar el tipo de unión, esta información actualmente no se utiliza para la deducción ni para la isomería. Esto es en virtud de que no hallamos una única normativa en la literatura. En cambio para poder calcular la masa atómica y el lema químico las uniones y anidaciones se tienen en cuenta correctamente. Para hacer esto la fórmula se re-crea internamente en forma simplificada mediante objetos anidados.

Isómeros

Por ejemplo **C7H6O2** (*aldehído salicílico*) es isómero con el (*ácido benzoico*) lo cual es reconocido como si fuese una palabra de la 'familia' o similares, es decir si alguien ingresa: **C7H6O2** o tal vez: **C14H12O4** esto se simplificará como **2(C7H6O2)** y será reconocido como isómero tanto del *ácido benzoico* como del *aldehído salicílico*.

En el ejemplo anterior, el ingreso indicado será reconocido como ambos productos, con igual probabilidad de 0.50. Se los etiqueta a ambos como alternativas, con una bandera

'flag' de corrección ortográfica del tipo: Isómero-Químico del **C14H12O4**. Si se ingresa al analizador esta fórmula, se obtendrá lo siguiente (el nombre del isómero en este caso no se muestra y aparece como sinónimo dentro de la estructura):

```
ácido benzoico NCMSsQ<*H6C7O2;&C7H6O2;mass:122.1g/mol;Molécula>
```

También en este caso se observa que el sistema dedujo y armó una raíz morfológico-química '***H6C7O2**;' y un lema-químico '**&C7H6O2**'. Este lema no posee el mismo orden y por lo tanto, no es comparable con otras fórmulas. Por otra parte, la raíz se ha ordenado para permitir su búsqueda en la base de conocimiento interno. De este modo, puede encontrarse el elemento llamado: *ácido benzóico*, cuya raíz química (*isomérica*) coincide con la fórmula ingresada.

Compuestos Químicos y Aleaciones de Metales

Hay dos clases de símbolos que asumimos para uniones especiales, elegidas por la literatura para indicar compuestos y aleaciones:

```
"+": compound
"*": aggregational (like hydratation)
"#": aleation
```

Por ejemplo ciertas sales complejas y compuestas e hidratadas, como el talco, cuya fórmula sería algo así como **(MgO)3+(SiO2)4+H2O**

Mientras que el yeso **2CaSO4*H2O** se trata de *sulfato de magnesio hemi-hidratado*, nótese que en estos casos el número **'2'** antecediendo al Calcio **'Ca'** significa 2 moléculas enteras, y no dos átomos de calcio, en otras palabras esta formulación sería equivalente a escribir: **2(CaSO4)*H2O** haciendo uso de los paréntesis para clarificar. También se podría colocar así: **(CaSO4)2*(H2O)1**

Todo esto es reconocido correctamente por el analizador morfológico y reconocedor de entidades nombradas (N.E.R.) en forma automática y nativa, sin instruirle nada en lo absoluto.

Aleaciones Metálicas y su Estequiometría

En el caso de ciertas compuestos y aleaciones metálicas se permite colocarlas de este modo, indicando las partes proporcionales (*en peso o volumen*) por ejemplo: **10%Fe#90%Cu** indica una aleación de **10% Fe** (*Hierro*) y **90% Cu** (*Cobre*), el signo de unión se utiliza es **#** indicando aleación metálica, a diferencia de **+** que indicaría un estado de agregación indeterminado.

Esta formulación solo se permite usar en forma interna (*para las bases de datos ampliables*) y es entregada por el sistema al analizar nombres de aleaciones que conoce como constantán, alpaca entre muchos otros, estando los más comunes y frecuentes contenidos en la base de datos interna.

Elementos de la Tabla Periódica

También se incluye algunos conocimientos mínimos de metalurgia y física básica, partiendo de datos propios de los elementos químicos de la tabla de Mendeleïev, por lo que puede 'deducir' por inferencia (adrede *no se utilizan algoritmos avanzados para esto*) pero aún así y en forma básica puede resolver cosas basadas en su fórmula y su conocimiento sobre los componentes, por ejemplo puede deducir si un compuesto 'desconocido' es orgánico o no, si contiene metales, si es tóxico, si es una aleación, en cambio sí podrá calcular exactamente su masa molecular, en algunos casos sabrá el año de su descubrimiento y su/sus descubridor/es, sus puntos de fusión y ebullición, si es peligroso por ser radioactivo, tóxico y algunos datos más.

Ejemplo de uso

Veamos que obtendríamos para el titanio (**`Ti`**) pues hay una inmensa cantidad de información, incluyendo su nombre en diversos idiomas, datos químicos, físicos, de origen, quien lo descubrió y cuándo, etc.

entrada:

```
Ti
```

salida:

```
NCMSsQ<*Ti;atomic_number:22; period:4; group:4;
discovered_by:Gregor_y_Klaproth; year_discovered:1791;
cristall:hexagonal; family:transition_elements; state:solid;
oxidation:+3_+4; rayon:1.32/1.47/-; ionization:6.82;
electroneg:1.54; elecconf:[Ar]_3d2_4s2; de:Titan; it:titanio;
fr:titane; en:titanium; ca:titani; es:titanio; mass:47.88g/mol;
density@20ºC:4.5g/cm3; boil:3287ºC; fusion:1668ºC;
Metal_de_Transición; Sólido >
```

Otro ejemplo indicará todo lo que el sistema puede saber o "ver" a partir de cosas, como ser un simple conjunto de 3 letras: **`HCl`** indicando una fórmula química simple.

entrada:

```
HCl
```

salida:

```
NCMSsQ <*HCl;&ClH; mass:36.46g/mol; Molécula; =HCl;
=ácido_muriático; =ácido_clorhídrico; =salfumán; =clorhídrico>"
```

> Esto dice que es un *Nombre Común Masculino Singular* con una clasificación Semántica de *'Substancia Química*' que a su vez tiene propiedades de ser: Molécula, posee 4 nombres alternativos, también llamado sinónimos: 'ácido_muriático', 'ácido_clorhídrico', 'salfumán' y 'clorhídrico'; posee una masa atómica-molecular *'mass'* de 36.46 g/mol. (*origen de los datos: Tratados de Química Analítica Clásicos como el Mahan, incluyendo Wikipedia*)

Apéndice 5

Explicaremos la necesidad y el concepto de reconocer determinados conjuntos de palabras y símbolos de una oración o sentencia de entrada, identificando claramente determinadas 'cosas' que llamaremos "entidades nombradas" y que poseen un significado único como un todo, por lo general diferente al de sus componentes.

Entidades Nombradas

Estas entidades existen en diversos idiomas, son de muchos tipos y su clasificación es extremadamente variada, compleja y dependiente del contexto; se ha decidido reconocer cierto tipo limitado de entidades de uso frecuente, las demás pueden anexarse mediante diccionarios y mecanismos en etapas posteriores de análisis del texto.

Un poco de Teoría

Cuando se analizan textos, hay una frecuente confusión entre una palabra, un término, una abreviatura, un acrónimo, una fecha y hora, una unidad, una fórmula matemático-química y su *función gramatical*, además de la interpretación correcta de su significado.

Una palabra escrita se podría definir como una secuencia de letras y sus separaciones con las demás palabras podrían ser los espacios, siendo a veces también ciertos signos de puntuación como también el principio y fin del escrito o frase. En consecuencia una palabra sería algo como: "palabra" “prodestricks” “üderlaetung” “thornshkist” “kaput”

Ahora que definimos una *palabra*, definiremos su función en un escrito, esto por lo general se llama función gramatical o sintáctica (*por sintaxis, del griego que significa relativo a las posiciones*). Esta función es muy conocida y tiene que ver con una clasificación asociada al tipo de función dentro de una unidad semántica (*de semios: significado*).

Función Semántica

Entre estas clasificaciones hay diversas jerarquías, entre las más básicas hallamos los sustantivos, los verbos, los adjetivos y los adverbios, además de otras partículas conectivas como las conjunciones, las preposiciones, los determinantes y las interjecciones; también hay algunas palabras extra como "*comodines*" que funcionan tomando el lugar (*su significado*) de otras, llamadas pronombres.

Hay otro nivel de clasificación sintáctica que corresponde a partes de la oración como sujeto, predicado, complemento, complemento agente, objeto directo, objeto indirecto, circunstanciales, etc. No entraremos en esta disquisición puesto que es un tema de lingüística y no corresponde a los objetivos ni conclusiones de esta tesis.

Existen funciones muy generales como las de un determinante, el cual lo cumplen muchas de estas categorías como los artículos, algunos adjetivos, algunos pronombres y las preposiciones; todas ellas ayudan a definir algo, agregándole información de deixis

"sentido, indicación" a la palabra que preceden, la cual suele ser un elemento nombrable, o nombre.

Resulta que si la 'cosa' nombrada es una sola palabra, ésta suele ser un sustantivo, el cual también puede ir "adornado" de algún adjetivo o construcción adjetiva, cuya función (*y su significado*) es que modifica al nombre.

Por ejemplo:

Las niñas alegres

En este caso 'las' es un artículo determinante, 'niñas' es el sustantivo núcleo del sintagma y 'alegres' es un adjetivo postnominal, actuando como modificador del núcleo.

En cambio si ponemos una sección como esta:

El Banco del Río de La Plata brinda préstamos

Banco del Río de La Plata es una Entidad Nombrada, con toda su significación asociada, la cual suele ser regionalmente diferente. En Argentina significa una entidad financiera conocida, tal vez en España o en Perú, ésta no exista y sea una dudosa definición.

Pero la lengua es así, es un emergente de la sociedad y su interpretación es bastante variada, en especial en diferentes regiones y contextos, por lo cual la regionalización del comportamiento de un sistema es tan importante como para una persona saber el idioma y acepciones de cada sitio para lograr comunicarse correctamente.

Los sistemas de procesamiento de lenguaje, permiten por lo general, reconocer bajo contexto éstos y otros grupos de palabras cuya función es única, basado en diccionarios y heurísticas de reconocimiento variadas llamadas justamente ***Named Entity Recognition*** o su abreviatura ***NER***.

Clasificación

Como se ha mencionado, los grupos de palabras ‘indivisibles’ por su función semántica o gramatical, se dividen en varios grupos.

Se enunciarán algunos de los grupos en que se pueden clasificar las entidades nombradas, en orden descendente a su contenido gramatical y conteniendo de abstracción y simbolismos.

Entidades Nombradas (Named Entites)

- Locuciones
- Términos
- Nombres Propios
- Abreviaturas

- Acrónimos y Siglas
- Fechas-Horas
- Unidades
- Números
- Fórmulas Químicas

Existe un segundo nivel de clasificación basada en el tipo semántico de la entidad, que no se tratarán en este trabajo como ser: si son un lugar geográfico, se refiere a una persona, se refiere a una fecha, una hora, una entidad matemática, etc.

A continuación se explicará brevemente algunas de las clases enumeradas anteriormente, con algunos ejemplos asociados, puesto que las otras ya están en Apéndices especialmente dedicados.

Locuciones

Es un grupo de entidades simples, a las cuales el 'uso' les ha conferido uno o más significados especiales y funciones gramaticales específicas, muchas veces de orden metafórico y no coincidente con el análisis del sentido de sus constituyentes. Por ejemplo, vemos un grupo de palabras como ***a cada rato*** en una oración como ésta:

> ***El tren pasa a cada rato***

Estas tres palabras en verde, funcionan como una única función gramatical y su función gramatical y semántica es un Adverbio Modal-Temporal y también en última instancia puede funcionar como una interjección usada de este modo:

> ***¡a cada rato!.***

Si empleamos el lematizador los y lo analizamos, los unirá en una sola 'cosa' o entidad llamada locución y las etiquetará como adverbio y/o locución.

```
a_cada_rato:
RG    (Adverbio General)
I     (Interjección)
```

Ejemplo de Locuciones

> ***por lo tanto, tal cual, por las dudas, tal vez, en caso de, etc.***

Abreviaturas

Dr. Sr. Ing.

Las abreviaturas son como lo dice su significado, palabras muy frecuentes abreviadas para no sobrecargar la lectura.

Se las reconoce tanto con los puntos finales, como sin los puntos; considerando su falta como un "leve" error de ortografía e indicándolo en un flag; salvo que la sigla o abreviatura no exista y haya una palabra exacta en el diccionario que la remplace, en este caso se presentan ambas como alternativa.

Acrónimos

Dr. Sr. Ing. S.A. S.R.L. SACI.

Las abreviaturas y acrónimos, se reconocen tanto con los puntos como sin los puntos, considerando su falta como un "leve" error de ortografía, salvo que la sigla o abreviatura no exista y haya una palabra exacta en el diccionario que la remplace, en este caso se presentan ambas como alternativa.

Ejemplo de acrónimos:

RRHH, RAM, TTL,

Se reconoce una lista limitada de acrónimos, del orden de 900, éstos se pueden o no expandir a las palabras que lo conforman, mediante un ‘flag’ del lematizador.

Términos

Ahora bien, como la plataforma ayuda al análisis del texto de entrada tratando de clasificar las "palabras" escritas por el usuario, de modo gramatical, pues hay un dilema, ciertas palabras funcionan como "grupo" y si significado dista de la interpretación combinada de sus significados individuales. Esto también ocurre con otros conjuntos de palabras llamados términos, por ejemplo

"pájaro carpintero"

Significa "pájaro" = un ave + "carpintero" = un oficio humano; pero esto resulta en una contradicción, o al menos yo no conozco ningún ave a quien le podría encargar un mueble!

Sin embargo hay un ave llamada "pájaro carpintero" y este conjunto de palabras está tomado como una sola "cosa" y esa entidad funciona como un sustantivo común el cual resulta estar en singular y masculino; pues bien: esto es una "entidad nombrada" del tipo "término".

Si analizásemos la siguiente oración:

En el Banco del Puente de Bilbao se otorga un crédito interesante.

Llegaríamos a conclusiones ridículas tratando de entender que cosa es una oración formada por estos grupos de palabras, pues hay muchas acepciones para el significado semántico de ***banco*** (*silla sin respaldo* o una entidad financiera, o el verbo ‘bancar’ en

primera persona singular del presente indicativo) luego '***del puente***' constata que está ubicado en un ***puente*** de la ciudad de ***Bilbao***.

Sería ridículo pues, el poder obtener *créditos interesantes* de *un tipo de silla sita en la plaza de una cierta ciudad*; pero en realidad "*todo este conjunto*" que recita: "*Banco del Puente de Bilbao"* representa una única "cosa" y es justamente una ***entidad nombrada***.

Detalles de Implementación

Se incluyeron un número de alrededor de 1300 locuciones, cuya fuente es la *Real Academia Española*, agregando manualmente alrededor de 300 necesarias para cubrir algunas abreviaturas usadas en Argentina y el resto de Latinoamérica. Se incluyeron también alrededor de 200 términos comunes, mayormente del ambiente de biología, botánica y zoología. Su listado no se incluye, dado que no aportaría nada, siendo parte de un recurso externo a esta tesis.

Clasificación Lingüística

Presentamos la clasificación con algunos ejemplos para inferir y/o poder entender el grado de dificultad de su análisis. El clasificador es parte del analizador morfológico y lematizador robusto. Se enuncia este listado con ejemplos dada su gran variedad.

Palabras Gramaticales

Verbos:

Infinitivos: *correr, decir, cantar*

Gerundios: *corriendo, diciendo, cantando*

Participios: *corrido, dicho, cantada*

+conjugados: *(yo) corría, (nosotros) decimos, (ellos) cantaban*

Con enclíticos: *(él) decíanoslo*

Adverbios: *rápidamente, sosegadamente*

Adjetivos: *amarillo, despacio, rabioso*
+flexiones: *amarillento, despacito, rabiosísimo*

Sustantivos: *perro, piedra, auto, camión*
+flexiones: *perrito, piedrotas, autos, camionetitas*

Pronombres Posesivos: *nuestro, vuestro, suyo*..

Pronombres Personales: *yo, tú, él, ella, nosotros..*

Pronombres Demostrativos: *eso, esto, aquello, aquel..*

Pronombres Interrogativos: *qué, cuál, quién, dónde, cuándo..*

Preposiciones: *más que cual quien cuyo donde cuando como*

Conjunciones: *que y o u e &*

Determinantes: *el, la los, lo, este, aquel, sendos*

Números

Ordinales: *primero, segundo, décimo, vigésimo..*
Cardinales: *uno, dos, quince, veintisiete..*
Partitivos: *un medio, tres cuartos, dos décimos*
Multiplicativos: *doble, triple*

Nombres Propios de persona (*given names*)
(algunas suelen estar en diccionario)

Andrés, Jorge, Rómulo, Juan, Pedro, Gabriela..

Nombres Propios puros, geográficos y apellidos
(rara vez estan en diccionario, sólo aparecen en libros y padrones)

Keops, Hohendahl, Gonzalez, Ramadán, Georgia, New York

Palabras Importadas de otros idiomas:

garçón, cinema, revue, video, high-fidelity, whisky, überwachung

Palabras-Basura (impronunciables):

jlsfunsukhjesdlk, hjiofj, gherrtoik, jhyg52gh962hs

Palabras fuera de vocabulario e idioma:

Abshaltungen, isopterodinosaurópodos , precolpasiado

Signos de Puntuación:

. , ; … ¶ §

Símbolos Matemáticos:

Unitarios: + - ±
Operaciones Binarias + - * / ^
Operadores: Λ Δ Σ Π Ψ Φ
Comparación: ≠ ≤ ≥ ≡ ≈ : = « »
Especiales: α β γ δ ε ζ η θ ∞ Ɛ ß μ π

Expresiones Lógico-Matemáticas:

$$½\ x^2 + ¾\ y^3 \leq \ \Sigma\ e^{\pi i\ /\ (2\ +\ k)}$$

Signos agrupadores, con diferenciación entre apertura y cierre

Pregunta: ¿ ?
Exclamación: ¡ !
Comillas: " ‘ ’ “ ”
Paréntesis, Corchetes y llaves: *( ) [ ] { }*

Símbolos Especiales

Copyright: ©
TradeMark: ™
Registrado: ®

Separadores de Listas: *, : : / |*

Símbolos que representan Monedas y/o Unidades

¢ $ £ € Pts ¥

Abreviaturas de países (IANA ISO)

AR, US, ES...

Abreviaturas de Moneda (ISO4127)

ARS, USD, EUR, FRA...

Símbolos que representan fracciones:

⅝ ⅓ ⅔ ¾½

Símbolos que expresan Potencias:
x² y³ z¹ pº

Numerales Especiales como Ordinales:
1ª 2º

Porcentajes:
20.3%

Unidades Sexagesimales con notación especial

10° 15' 20"

Acrónimos Varios y Siglas (frecuentemente no están en los diccionarios)

RIIAA.
Fi-UBA.

MP3
3.5G
PNP
JFET
N-MOS

Nombres y abreviaturas de Unidades

Ω Kg A µF °C N cm m²

Números en diversos formatos:

Científico: +1.23 E-10
Hexadecimal: 0xFF31A
Binario: 10010110110
Romano: MCMXII

Caracteres Especiales:

Guión Bajo: _

Fórmulas Químicas (Normalizados bajo formatos: IUPAC, etc.)

$H2O \equiv NO_3\text{-}NH_4 = 2(COOH)$

Direcciones IP de internet

192.168.0.125

Direcciones URL y URI

http://web.fi.uba.ar/~ahohenda
www.pbox.com.ar
ftp://web.fi.uba.ar/data?id=2344&p=a%20b

Direcciones de e-mail

andres.perez23@gmail.com

Hashtag de Twitter

#HoyNoHayComidaBuena

Dirección de Twitter

@andyhohen

Tiempos y Horas del Día

2:12hs

10:30 PM
10 hs. 20' 32"
3 y cuarto
dos y media

Fechas y Fecha-Hora
10-oct
20/4/2011
1º Nov. 2003
mediados de noviembre del año pasado
viernes que viene
jueves 3 de agosto del año 2005

Importante

Todas estas partículas *'tokens'* y varias más que no se han incluido para no alargar este escrito, deben ser "segmentadas" antes de ser tratadas por un sistema que las pueda identificar y entregar a la etapa posterior con un mínimo análisis y un etiquetado correcto.

Algunas de estas partículas *'tokens'*, como ser las fechas / horas y los números dichos con palabras, son detectadas separadamente en etapas posteriores, dado que las palabras constituyentes podrían tener errores y también por la ambigüedad misma que sean tratadas en esta temprana etapa.

Concluyendo sobre este tipo de detección llamado Reconocimiento de Entidades Nombradas (del inglés: NER *Named Entity Recognition*), no constituye nada novedoso en esta tesis, simplemente consiste en trabajo de programación e implementación minucioso con algoritmos acorde a cada cosa.

Apéndice 6

Diccionarios y sus Antecedentes

El diccionario de la *Real Academia Española* (*RAE*) en su edición XXI posee cerca de 90 mil términos, repartidos aproximadamente parejo entre verbos, adjetivos y sustantivos, con un número menor al 5% de adverbios y menos número de otras partículas como preposiciones, pronombres y conectivos.

Resulta que este diccionario, al igual que la mayoría, no contienen las palabras en plural, en femeninos, los verbos conjugados, los diminutivos, aumentativos, peyorativos, los adverbios usuales derivado de los verbos como 'apropiadamente' derivado del verbo apropiar. Tampoco contienen ninguna combinación de estos "efectos" que se llaman en lingüística 'derivación' y 'flexión'.

Antes de saltar de lleno en la definición de un diccionario capaz de reconocer toda forma morfológica de una palabra en un idioma, vamos a analizar los tipos gramaticales de palabras y la variabilidad que presentan.

Los Verbos

Los verbos en español, se conjugan de 50 maneras diferentes, según el diccionario de conjugación de Larousse, el cual trae estos modelos de conjugación para que cada uno se imagine como conjugar un verbo cualquiera, con esa plantilla. El listado de los verbos clasificados apenas asciende a 5000 y hay algunas aplicaciones informáticas que reproducen estas conjugaciones, mas son comerciales, limitadas para un ***iPad / iPhone***. Esto claramente no cubre las necesidades de información.

Para conjugar un verbo se tienen 7 modos de personas (*yo tu vos el/ella nosotros vosotros ellos*) luego hay 3 tiempos: pasado, presente y futuro, combinados con modos como el indicativo, subjuntivo, condicional, e imperativo; con variantes de perfecto e imperfecto.

Además de los pronombres átonos y enclíticos como 'se' 'lo' 'me', aplicados en los verbos pronominales como secar**se** dormir**me** etc. esto da un promedio de 160 a 200 conjugaciones y variaciones para cada verbo, si ponemos esto junto con 50 modelos de conjugación en donde casi todos los verbos más frecuentes, son irregulares.

Haciendo las cuentas muy simplificadas si consideramos 30.000 verbos diferentes por 200 conjugaciones, da $6x10^6$ de formas conjugadas, que es una cantidad bastante grande.

Sustantivos y Adjetivos

Ahora analicemos aproximadamente los sustantivos y adjetivos, ambos poseen por lo general 2 números (plural y singular), 2 géneros (masculino y femenino), pero en ambos hay aumentativos de diverso tipo, tanto como peyorativos y diminutivos, llegando a

entre 20 y 50 formas diferentes si incluimos las variantes de los países de habla hispana; esto aplicado al número de sustantivos y adjetivos totales da 60.10^3 x 50 = 300.10^3

Hasta aquí el total de palabras que tendríamos, sumando las 100.000 iniciales, es alrededor de 6.4 Millones de palabras diferentes, y esto es una gran cantidad.

Prefijos y Sufijos

El problema no termina aquí: existen en español, una cantidad importante de prefijos muy frecuentemente usados, a la vez que sufijos de índole médica, química y científica.

El número de prefijos es de alrededor de 1000 y muchos de ellos son combinables entre sí, es decir se aplican unos y luego los otros, frecuentemente más de una vez.

Este proceso al igual que la conjugación es licito aplicable a toda palabra como adjetivo, sustantivo, adverbio e inclusive a un verbo. En algunos casos modifican ligeramente las palabras en el punto de unión, por temas relacionados con la fonética entre otras reglas.

Este proceso éste se llama ***derivación morfológica*** y lo usamos a diario, casi sin saberlo, cuando decimos por ejemplo:

"esto está *recontrabuenísimo*"

Una persona lo entendería fácilmente, en cambio un diccionario-corrector clásico como el del MS-Word y la mayoría, por no decir todos los demás procesadores de texto del planeta, que jamás la interpretarán correctamente, ninguna otra palabra del estilo.
Por eso es que se le agregan parches, llamados diccionarios “de usuario” para permitir estos términos, que el pobre usuario debe agregar manualmente.

Palabras Parasintéticas

El problema que suscita con los prefijos y sufijos, es que si asumimos apenas unos 200 prefijos son de uso frecuente (*dejando los otros 800 de lado*), el número de palabras asciende a 1260 millones de palabras, i suponemos una sola anidación.

En cambio si aceptamos apenas 25 prefijos sean usados combinados (*anidables dos veces*), tendremos una cifra de alrededor de unas 6 mil millones de palabras diferentes.

Este fenómeno de armar palabras nuevas se llama o parasíntesis de palabras y las palabras resultantes son palabras ***parasintéticas***

Hay estudios sobre parasíntesis del español, que revelan que en el caso de textos del español, se han contado y estimado un universo de alrededor de 3000 millones de palabras diferentes, sintetizables [12]. Esta cifra condice con el orden de la estimación hecha en esta sección.

Este problema no termina aquí y se agrava con las palabras fuera de diccionario, las importadas y aceptadas en otros idiomas, lo cual constituye otro gran problema.

Búsquedas en Diccionario

Cuando se realiza una búsqueda en un diccionario electrónico, se esta en un problema similar o equivalente a una búsqueda en una base de datos de palabras.

Hay muchas maneras de hallar una coincidencia contra una lista de palabras en la base de datos; la más simple es comparar palabra por palabra. Esto se llama búsqueda lineal y si hay más de algunos miles de palabras ya esto es inviable, por el tiempo que lleva y el espacio de memoria de proceso inmovilizado (*estático*) necesario.

Indexación

Para esto hay una disciplina de la informática dedicada exclusivamente a esto y es la indexación de datos, que favorece los mecanismos de búsqueda.

Entre muchas teorías desarrolladas al respecto podemos rescatar la de un sistema de almacenamiento en un árbol, perfectamente balanceado; el número de accesos a memoria o disco, necesarios para hallar un elemento de una número N de datos (*previamente ordenados o indexados*) es del orden de ***log(N)*** en donde la base del logaritmo depende de la cantidad de hojas del árbol, frecuentemente usamos 10.

Esto nos permite calcular el número promedio de accesos a disco o memoria para hallar una palabra en 1 millón es de alrededor de 6. Esto resulta (*con discos rápidos, de 10k RPM*) en una velocidad de acceso promedio de alrededor de ***30 mS*** por búsqueda, asumiendo tiempo cero de proceso.

Si las palabras en la base, fueran 3 mil millones, el tiempo se elevaría a ***270 mS***, cerca de ¼ seg. Esto es completamente inaceptable para este trabajo, pues para probar apenas 20 opciones, se tardaría 5 segundos!

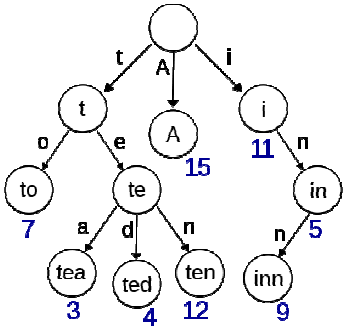
Como conclusión de esta especulación, el poseer una base de palabras tan grande, es impráctico y resulta inactualizable, pues requeriría un espacio de disco de varios terabytes *(caro e impráctico hoy, al menos en el 2013*) y además no se podría hacer, pues esta lista de miles de millones de palabras, simplemente no está disponible.

Apéndice 8

Métodos de Búsqueda con Strings

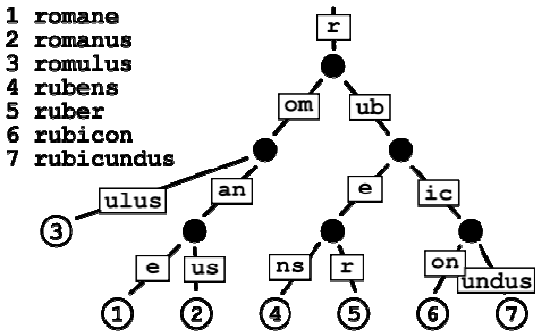
Se presentan algunos de los métodos utilizados para indexar datos en forma de strings, con ejemplos visuales muy simples, pues hay muchísima literatura al respecto. Esto no pretende ser completo ni exhaustivo y solo presentan algunos métodos junto a un ejemplo simple en forma de gráfico.

Trie



Es una estructura de árbol basada en posiciones de las letras dentro de la string, es poco eficiente en memoria pero muy veloz para su acceso. Su complejidad o costo de proceso es como máximo **`O(n)`** donde **`n`** es el número de letras de la string a hallar.

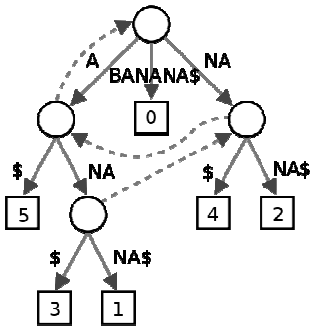
Patricia Trie



Es una mejora del **Trie**, compactando grafos los lineales del **Trie** en 'substrings' y logra una mayor compaticidad de memoria guardando una velocidad similar al del **Trie**. Su costo de proceso es como máximo **`O(n)`** donde **`n`** es el número de segmentos de la string, siendo menor al número de letras, dado a que condensa substrings y en consecuencia el costo final es dependiente de la implementación del algoritmo en cada plataforma.

Suffix Trie

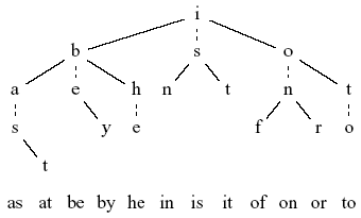
Es una versión que almacena punteros a todos los sufijos, permite una búsqueda rápida de partes de una palabra pero es muy ineficiente en términos de espacio de memoria.



Su costo de proceso es como máximo **`n*O(n)`** donde **`n`** es el número de segmentos de la string, dado a que busca todos los substrings y al igual que casi todos estos métodos, el costo final es dependiente de la implementación del algoritmo en cada plataforma.

Ternary Search Tree (TST)

Este es un árbol con tres caminos, el cual permite, mediante una simple adición de un puntero hacia arriba, convertirse en un sistema de búsqueda aproximada. Lo utilizamos por esa ventaja, en el sistema de búsqueda fonética aproximada.



Apéndice 7

Opciones del Analizador Morfológico

Listado de opciones del Analizador Morfológico Robusto al que llamamos, en el cual se han incluido los algoritmos de reconstrucción de errores del presente trabajo.

Este Lematizador reconoce secuencias de caracteres de casi todo tipo y hace el mejor esfuerzo para detectar los componentes léxicos.

En el caso en que no encuentre sus características en su diccionario interno, el cual de hecho posee una gran cantidad de términos, los cuales exceden el uso cotidiano del lenguaje en cuestión; el sistema analiza minuciosamente cuáles alternativas hay conforme a todas las estrategias disponibles, listadas en el siguiente conjunto de 'flags':

Uso de Diccionarios

UseStandardDict
UseExternalDict

Sección de Lematizado

CheckPreffix
CheckSuffix
CheckRoot
CheckInfix

Corrección de Ortografía

UseSpellCorrection

Reconocimiento

OnlyValid

Reconstrucción Fonética

PhoneticMatch

Etiquetado de palabras desconocidas

UseHeuristics
UseMorphoGuess
UseSuffixGuess
UseFreelingGuess

Reconocimiento Numérico

FloatSciNumbers
MultiWordNumbers
WordNumbers
ColloquialNums
RecognizeRomanNumbers

Reconocimiento de Moneda, Números, Unidades, Química y Fechas

RecognizeMoney
RecognizeUnits
RecognizeChemicalFormulas
RecognizeDates

Reconocimiento de Locuciones

RecognizeLocutions
MergeLocutions

Abreviaturas

RecognizeAbreviated
ExpandAbreviated

Los 'flags' están en inglés en forma abreviada y usando PascalCasing para diferenciar las partes ya que al usarse como nombre de variable enumerativa lógica en C# no se pueden incluir signos de puntación dentro de ellos.

Glosario

En esta sección se explican algunos términos específicos y que son importantes de entender en detalle para comprender este trabajo. Se describen lo suficiente, tratando de enunciar las principales ventajas y limitaciones prácticas en su utilización actual y, en donde sea posible, se cita el estado del arte o las "prácticas frecuentes" en el tema.

Analizador Morfológico

Es un sistema capaz de reconocer y etiquetar partes constitutivas de un texto cualquiera. Por lo general, los que hay comercialmente, poseen léxico acotado (poco abarcativo), buscan exactitud, no hallan información más allá de la incluida puramente en forma "dura" en sus bases de datos, son "duros" y responden pobremente ante ambigüedades. Debieran de tener: Léxico amplio y extensible, Robustez, Capacidad de inferencia, Corrección Ortográfica y Fonética, Análisis de alternativas, Manejo de términos multi-palabras y Abreviaturas, extraer información aproximada, ante la falta de datos, inferir significados, manejar palabras parasintéticas con criterio robusto, Identificación de Idiomas, Detección de pronunciabilidad de supuestas palabras para descartarlas y ver las como ruido 'noise', Detección de formulismos: matemáticos, químicos, computacionales entre muchos otros.

Chunkers

Son programas que seccionan un texto en partes (llamadas *chunks*), suelen ser más robustos que los parsers, aunque suelen fallar ante NER y como efecto colateral, producir colisiones, dependiendo del orden de aplicación del método, es difícil definir los límites de cada chunk, se suele usar un balance de máxima verosimilitud con alguna función de peso general o gramática subyacente. Debieran de no manejar límites definidos y poder recibir precisiones de las etapas posteriores como NLU/NER.

Forma Escrita

Es toda combinación posible de un número determinado de caracteres permitidos por un lenguaje en cuestión. Por lo general la cantidad diferente de posibles formas escritas es un número grande y responde a un cálculo combinatorio, dado el número de caracteres diferentes N_C y el N número total de caracteres de la forma escrita, es posible escribir $N_C{}^N$ formas escritas diferentes. Se distingue del término **palabra** en que no contempla ningún otro rasgo como ser su existencia en diccionarios, pronunciabilidad, etc.

Languaje Natural

Traducido del inglés (*Natural Language*), parece trivial la definición, pero define al texto como lenguaje en sí y basado en su contenido de información interpretable por un humano. En otras palabras es el texto tal cual como se escribe, redacta y habla, pero referido a lo que contiene. En especial está definido como lenguaje natural todo aquello que tienda a tratar al texto por su significado, función lingüística, contenido, tratando de alcanzar la comprensión tal como un humano lo haría. Esta ciencia es muy compleja, altamente interdisciplinaria e incluye mucha matemática, estadística, algorítmica, lingüística e ingeniería electrónica y de software. Interviene fuertemente en todos los métodos de reconocimiento de voz (ASR) Síntesis de Voz (TTS), en minería de textos (IR) y búsquedas en la web (Google), sistemas de diálogo, etc.

NER

(NER: Named Entity Recogition = Reconocedor de Entidades Nombradas) es un sistema capaz de agrupar términos y siglas, identificando grupos que significan cosas como un todo, el cual muchas veces no condice con un análisis del significado de la construcción gramatical completa, tal es el caso de fórmulas matemáticas, fórmulas químicas, locuciones, nombres de compañías, bancos, números, fechas y horas dichas en forma de palabras y números mezclados de cualquier modo, etc. Los mecanismos existentes para determinar estas entidades son muy variados y dependen del contexto y de la aplicación o necesidad, suelen usar mezcla de reglas, con aprendizaje automático y heurística.

Trabajan sobre las estructuras previamente Tokenizadas, suelen usar algoritmos de aprendizaje automático + heurísticas 'duras'. Andan razonablemente bien pero no abarcan conceptos. Debiera de hacerse desde la semántica hacia atrás (buscar cuando una frase no tiene un significado claro, cual puede ser una NER que aclare todo)

NLP

(*Natural Language Processing* = Procesamiento de Texto Natural). Es la ciencia que estudia el lenguaje natural (ver) y agrupa un conjunto de métodos, algoritmos y otras técnicas proveniente de varias áreas, especialmente desde la estadística y la inteligencia artificial.

NLU

(Natural Language Understanding = Comprensión de Texto Natural). No está muy claro que es esto, pues faltan modelos cognitivos únicos y con definiciones claras, los esfuerzos de reglamentar la web semántica como OWL, CyC y otros no proveen mas que 'repeticiones' tipo loro de los razonamientos pre-cargados. Debieran de tener una lógica mas blanda para poder evaluar mejor alternativas en contexto, no mediante lógica sino mediante otros modelos, no sé si usar combinación de lógica tipo fuzzy, redes neuronales tipo SOM o clasificadores estadísticos (SVM, tipo kernel, PCA, bayesianos, ANN, etc.). El problema reside en la lógica (¿existe eso?) del sistema.

NP duro (***NP-hard***)

Proviene del inglés: Number Permutations Hard, es cuando la longitud del proceso de un algoritmo que resuelve un problema depende de una magnitud combinatoria, ésta suele crecer sin límites y termina siendo irresoluble por la imposible cantidad de operaciones a realizar para completar el algoritmo, tornándolos del tipo irresoluble en tiempos polinomiales o razonables; en esto se basa la criptografía, como ejemplo.

NO-PALABRA

Se trata de formas escritas factibles de ser pertenecientes de un idioma, es decir contienen solamente los caracteres permitidos por éste, deben ser pronunciables y responder a criterios morfológicos y fonéticos del idioma, es decir permiten una correcta conversión a fonemas, factible se ser articulados. Otra condición suele ser que no sea una derivación o flexión de palabras existentes, mientras que una OOV o palabra fuera de vocabulario es más permisiva e incluye principalmente estas formas, candidatas a ser reconocibles por una persona e inclusive inferido su significado. En otras palabras, una no-palabra es una forma escrita (combinación válida de caracteres) pronunciable cuyo significado no pueda ser inferido o reconocida por constituyentes morfológicos.

Suelen ser candidatas a ser expresamente incluidas en diccionarios. De hecho cada tantos años la Real Academia Española, así como la Academia Argentina de Letras, entre otras muchas autoridades en letras, incluyen palabras nuevas, muchas de ellas importadas de otros lenguajes, inventadas o emergentes de la misma sociedad y de uso suficientemente frecuente y con uno o más significados concretos.

→ ***ver*** : **palabra** y **palabra fura de diccionario**.

NP-Hard / NP-Duro

Number Permutation Hard, Se llaman así a problemas que no tienen una solución única obtenible en un número acotado y/o polinomial de pasos o etapas; mayormente sus leyes son números combinatorios de algunas de sus variables, cuyo número rápidamente alcanza valores imposibles de explorar computacionalmente o por fuerza bruta. justamente son éstos algunos de los problemas típicos a ser 'atacados' mediante heurísticas y ciertos mecanismos de la inteligencia artificial.

GLN / NLG

Natural Language Generation, Generación de Lenguaje Natural

Es la generación de lenguaje basándose en datos y lógica gramatical específica (ATG).

Lo existente es radicalmente complejo, suele ser exclusivo para el inglés y no se obtienen resultados robustos, son mecanismos utilizando Planning (GraphPlan y sus variantes) como base, sintetizando estructuras con granularidad gradual, pero olvidan por lo general la semántica subyacente y la retórica asociada al diálogo.

OOV

Abreviatura usada en lingüística, proviene del inglés *Out Of Vocabulary*. Y significa Palabra fuera de vocabulario, → ***ver*** palabra fuera de diccionario

Palabra Fuera de Diccionario

Es una **forma escrita** no incluida expresamente en el diccionario del contexto, no es una definición global sino acotada a un contexto de análisis morfológico en determinado idioma. Debe reunir ciertas características como ser pronunciable y que pueda ser el resultado de una derivación o flexión morfológica a partir de una palabra existente, pertenecientes al diccionario del contexto. Esta derivación puede o no tener sentido semántico correcto, llamado sentido común, difícil de definir. Toda no palabra es una palabra fuera de diccionario, mientras que no toda palabra fuera de diccionario es una no-palabra.
→ ***ver*** : **no-palabra** y **palabra**.

Palabra

En el contexto lingüístico podríamos definir una palabra como toda forma escrita que satisfaga una de las siguientes características: pertenecer a un idioma, diccionario o a un listado de palabras aceptadas como tal. Se diferencian de las no-palabras en que deben necesariamente estar en uno de esos grupos y cumplir criterios mínimos de morfología y fonética.
→ ***ver*** : **no-palabras**, **palabras fuera de diccionario** y **OOV**.

POS

Abreviatura del inglés *Part Of Speech*: significa Partes de la Oración o Partes del Habla. Es el nombre que se le da comúnmente a la etiqueta o nombre de clasificación de las palabras basado en sus atributos lingüísticos y de corte netamente gramatical, como ser: sustantivo, adjetivo, verbo, si es singular, plural o neutro, la persona, su clase, el modo, el tiempo, etc.
→ ***ver*** : Apéndice 2, Etiquetas EAGLES son las etiquetas POS que utilizamos en la presente tesis.

POS Taggers

Término abreviado del inglés *Part Of Speech Taggers:* Significa sistemas etiquetadores de palabras basadas en su función gramática. Se etiqueta su tipo gramatical por su posición, determinando cual es la parte de la oración que le corresponde a cada palabra o partícula y resolviendo la ambigüedad de estas funciones, que es de 1.6 significados por etiquetas por palabra en promedio en español y de 2.4 en inglés. Son sistemas por lo general estadísticos y rápidos, suelen usar estadísticas aprendidas mediante bigramas o trigramas y resuelven con algoritmos de aprendizaje automático como Hidden Markov Models (HMM) usando el método de resolución por programación dinámica de Viterbi [32] y últimamente una evolución de éste, llamado CRF (*Conditional Random Fields*) de mejor alcance y precisión.
Todos ellos en general proveen una sola salida, siendo mucho más complejo que entreguen las N mejores. Se pierde información útil al clasificar "duramente" una palabra que por contexto es justamente 'la otra' = "la de menor probabilidad". Proveen una buena tasa de precisión y el estado del arte está cercano al 97%. Generalmente se atienen y sesgan al corpus usado en el entrenamiento y en consecuencia no generalizan bien pues dependen fuertemente de la estadística aprendida. → ***ver*** : POS

Parsers

Son piezas de software, que implementan algoritmos para determinar cuál fue la parte de una gramática que dio lugar a una secuencia de símbolos, o sea a partir de una secuencia de palabras de una oración, determinar sujeto, predicado, objeto directo, circunstanciales, y los núcleos uy modificadores de cada parte, etc.
Suelen ser autómatas determinísticos basados en tablas, del tipo push-down: LL, LR, GLR (este último es no determinístico). Son complejos, no hay gramáticas buenas ni simples (al menos para el español). Los LL y LR no manejan bien gramáticas ambiguas y se usan para computación en compiladores, el GLR es no determinístico y tiene a explotar exponencialmente con gramáticas grandes, Al ser rígidos (se basan en una gramática). Los parser del tipo link-grammar [Brill] (*grafos con proyección*) son mas flexibles pero la gramática es muy compleja de modelizar, tampoco son robustos ni garantizan un parsing óptimo.
Los de tipo HPSG son muy complejos de modelizar, aunque anda aceptable. Las TAGS son sirven para adquirir gramáticas mínimas, pero no demuestran ser útiles en parsing. Todos los estadísticos deben aprender una gramática de un corpus anotado, de lo cual hay poco y no de buena calidad ni lo

suficientemente abarcativos como para poder manejar un idioma en forma flexible y generalizada. Los modelos que hay no tienen aún en claro cual mecanismo/criterio de 'pruning' usar y como modelizar esas gramáticas, se buscan hacer hallando gramáticas mínimas, pero no siempre éstas son útiles para una tarea posterior como ser NLU, Q&A, etc.

Razonamiento y Modelos Cognitivos

Es complejo de definir, no está muy difundida esta rama de la ciencia, hay ramas como la lógica pura, la deductiva, la lógica de primer orden (FOL), sistemas de prueba de teoremas, razonadores automáticos, OWL y una gran cantidad de teorías, muchas completamente independientes con pocos lugares comunes. Lo que se quiere es lograr reglamentar y tal vez crear modelos de inteligencia humana. El querer imitar los sistemas mediante 'estadísticas' o imitaciones no es mas nuevo que el viejo engaño de la máquina 'robot' de ajedrez con un 'jugador dentro' del siglo pasado. Se destaca una nueva línea de teorías y trabajos con sistemas tipo quantum-logic (*Quantum Analogical Modeling* [R. Skousen]) no parece para nada descabellada, aunque emular este comportamiento sin un computador cuántico hoy, es NP duro e inviable, pero parece prometedor el concepto.

Tokenizador

Pieza de software que implementa un segmentador (cortadores en partes = *tokens*) de texto crudo (secuencia de caracteres o 'string') obteniendo segmentos como formas escritas, números, fórmulas matemáticas, químicas, etc.. Se usa siempre que se debe segmentar una 'string' en palabras, antes de usar analizador morfológico.
Por lo general son autómatas finitos, mayormente generados por software mediante un script escrito en alguna gramática regular. También se los llama **Lexers** que son basados en Expresiones regulares. Suelen hacer bien su trabajo aunque fallan fácilmente debido al "ruido" en el texto como secuencias no esperadas.
Un tokenizador apropiado para lingüística, debe ser robusto, tolerar errores y en ciertos casos proveer múltiples salidas especulativas (no-derterminista). Sin embargo es difícil crear este tipo de segmentador y que abarque toda forma escrita o secuencia de caracteres posible. Las soluciones adoptadas suelen formarse mediante una segmentación básica inicial, seguido por secciones capaces de realizar una unión de partes mal segmentadas y hasta realizar una re-segmentación basada en diccionarios y/o lógica e inteligencia posteriores. Esto es una tarea clásicamente de Procesamiento de Lenguaje Natural para determinación de segmentos que represeten términos o entidades nombradas → ***ver*** : NER
El trabajo de segmentado es sin dudas un problema NP-duro pues es de índole netamente combinatorio del número de caracteres del texto a segmentar, por lo cual toda solución tiene riesgo cierto de no terminar en tiempos polinomiales y por eso debe ser cautelosamente limitada.
Dado que el número de combinaciones recibibles es de orden combinatorio o indeterminado, la segmentación no constituyen una tarea trivial y por cierto las soluciones mayormente poseen fallas, Muchas veces estos segmentadores trabajan "uniendo paquetes de caracteres", que vienen de a uno por un canal que los transmite en forma contínua (*stream*) o del cual se desconoce a priori el número total de caracteres a recibir.