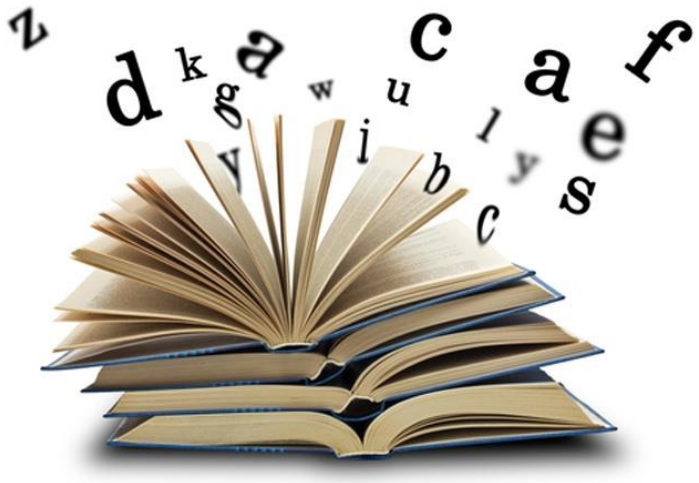

CORRECTOR ORTOGRÁFICO

Trabajo evaluación PLN I

Elaborado por:

- Inmaculada Perea Fernández
- Juan Raggio Pérez
- Eduardo Sánchez Karhunen
- Marina Sarmiento Pelegrina

Contenido

1 Introducción 3

2 Descripción del problema 3

3 Estado del arte 3

3.1 Ispell 3

3.2 Aspell 3

3.3 Microsoft Word 4

3.4 Google 4

4 Solución propuesta 4

4.1 API 4

4.2 Descripción 5

5 Prototipo 6

5.1 Construcción del diccionario base 6

5.1.1 Inicialización 6

5.1.2 Entrenamiento 7

5.2 Inclusión de nuevas transformaciones 8

5.3 Integración con diccionarios online 8

5.4 Exportación del diccionario base 9

5.5 Lista de sugerencias 9

5.6 Principales problemas 10

6 Propuestas de mejora 10

6.1 Niveles de revisión 10

6.1.1 Tipografía 10

6.1.2 Gramática 11

6.1.3 Estilo 11

6.1.4 Semántica 11

6.2 Distancia fonética y palabras homófonas 11

6.3 Aprendizaje 13

6.4 Asignación de pesos a las transformaciones 13

6.5 Cobertura léxica y tratamiento de palabras desconocidas 13

6.5.1 Conjugación verbal 14

6.5.2 Elementos compositivos y palabras derivadas (sufijos y prefijos) 14

6.5.3 Abreviaturas 14

6.5.4 Símbolos alfabetizables 14

6.6 Actualización 14

6.7 Detección de idioma 15

6.8 Inteligencia colectiva 15
7 Conclusiones 15
8 Referencias 15

1 Introducción

En este documento se presenta el trabajo realizado para resolver el problema propuesto en la asignatura PLN, en la que se planteaba idear un corrector ortográfico.

2 Descripción del problema

Se propone la construcción de un corrector ortográfico, es decir una herramienta que analice los errores de escritura que aparecen dentro de un texto, los notifique al usuario y dé la opción de corregirlos automáticamente o de ofrecer alternativas diferentes.

El corrector compara las palabras del texto con las palabras de su diccionario base y si las palabras del texto se encuentran en el diccionario, éstas son aceptadas; de lo contrario, el corrector propone términos similares.

Este proceso se puede realizar de forma automática, sustituyendo la palabra que se entiende como errónea, o semiautomática, es decir sugiriendo un conjunto de opciones posibles y dejando que el usuario elija cuál de las opciones es la correcta.

Los correctores actuales, además de comprobar la existencia o no de una determinada palabra en un diccionario, incorporan corrección gramatical y semántica. Comprobando si el texto cumple ciertas normas de la gramática (orden de las palabras, etc.) y de la semántica (frases que tengan sentido, etc.).

3 Estado del arte

El conjunto de herramientas disponibles es muy amplio, pero a continuación mencionaremos algunas de las más destacables o conocidas.

3.1 Ispell

Ispell es una herramienta de corrección para sistemas Unix que da soporte a un gran número de idiomas. Está basado en el cómputo de distancias de Damerau-Levenshtein sobre una transcripción fonética del vocabulario con el que trabaja.

Incorpora diferentes interfaces para ser usado como herramienta individual o con un editor como emacs.

Lee el archivo de entrada y cuando encuentra un error, genera una lista de posibles sugerencias que se muestra al usuario, para que este elija si se quiere modificar la palabra incorrecta por una sugerencia, ignorarla, o añadirla al diccionario base.

3.2 Aspell

Aspell es un corrector ortográfico de software libre diseñado para reemplazar a Ispell. Es el corrector ortográfico estándar para el sistema operativo GNU. También compila otros sistemas operativos similares a Unix y Windows. Da soporte a cerca de 70 idiomas.

A diferencia de Ispell, Aspell puede comprobar fácilmente los documentos UTF-8 sin tener que usar un diccionario especial. Aspell incluyen soporte para el uso de varios diccionarios a la vez y manejo inteligente de diccionarios personales cuando más de un proceso de Aspell está abierto al mismo tiempo. Sin embargo, Ispell sigue la convención Unix de ser un comando aplicado a un archivo, mientras que Aspell requiere otras opciones de línea de comandos. Los usos de la muestra incluyen:

- Ejecutar interactivamente a través del archivo_texto comprobar la ortografía (aspell check text_file).
- Permitir escribir una palabra (seguido de newline y Ctrl-D) para encontrar palabras que suenan igual (aspell soundslike).

3.3 Microsoft Word

Word automáticamente revisa la ortográfica y la gramática del documento en que se está trabajando. Si encuentra algún error, Word lo marcará con una línea de color debajo de la palabra. Si el error es ortográfico, la línea será roja. Si el error es gramatical, la línea será azul.

La forma que tiene Word para detectar las palabras erróneas es comprobar si las palabras de nuestro documento existen en el diccionario que lleva incorporado, junto con algunas reglas gramaticales. Lo que no es capaz de hacer Word, por el momento, es discernir el significado de las palabras en cada contexto, es el caso de las palabras homófonas.

La revisión ortográfica consiste en comprobar que las palabras de nuestro texto no son erróneas y la revisión gramatical trata de que las frases no contengan errores gramaticales como por ejemplo "Los libros son buenas"; donde no concuerdan el género del sujeto y del adjetivo.

Con Word podemos realizar una revisión ortográfica, gramatical o ambas a la vez.

3.4 Google

Google te muestra los errores ortográficos a medida que escribes. Para que Documentos de Google pueda encontrar los errores ortográficos, debes establecer el idioma. Las principales funciones de este corrector son:

- Comprueba la ortografía
- Añade palabras al diccionario para que no las detecte como errores ortográficos (también se pueden quitar palabras del diccionario)
- Muestra sugerencias ortográficas para las palabras mal escritas

4 Solución propuesta

Existen distintos enfoques para resolver el problema planteado, la solución propuesta en este trabajo está basada en el diccionario spell checker de Peter Norvig.

El algoritmo de Norving trata de encontrar las entradas del diccionario con la menor distancia de edición (distancia de Damerau-Levenshtein) del término de la consulta. Si la distancia de edición es 0, el término está escrito correctamente, si la distancia de edición es <= 2, el término del diccionario se usa como sugerencia de ortografía.

Generar todos los términos posibles con una distancia de edición <= 2 (elimina + transpone + sustituye + inserta) del término de la consulta y busca en el diccionario.

Para una palabra de longitud n, un tamaño de alfabeto a, una distancia de edición d = 1, habrá n eliminaciones, n-1 transposiciones, a * n alteraciones y a * (n + 1) inserciones, para un total de 2n + 2an + a-1 términos en el tiempo de búsqueda.

4.1 API

En este apartado se detallan las distintas funciones que componen el código de Norving definiendo lo que estas realizan:

- Función **words**: genera una lista de PALABRAS a partir del texto de entrada

- Función **P**: a partir de una palabra contenida en la lista generada devuelve la probabilidad de la palabra en el texto
- Función **correction**: tiene como entrada una palabra y como salida la corrección ortográfica más probable de dicha palabra
- Función **know**: tiene como entrada una lista de palabras y devuelve el subconjunto de la lista de palabras que aparecen en el diccionario de PALABRAS.
- Función **edit1**: tiene como entrada una palabra y como salida todas las transformaciones de esta que son palabras
- Función **edit2**: tiene que como entrada una palabra y usando la función edit1 toma de la salida de esta función aquellas palabras con una distancia de edición menor o igual a 2.

4.2 Descripción

La solución que propone Norvig está basada en el modelo de canal ruidoso (ver Figura 1). La palabra errónea se modela como si una palabra correcta hubiese sido distorsionada después de haber atravesado un canal de comunicación ruidoso. Este canal introduce ruido en forma de sustituciones, inserciones y otros cambios en las letras, haciendo que sea costoso reconocer la palabra original. El objetivo, por tanto, será modelar dicho canal.

El canal ruidoso es un tipo de inferencia Bayesiana. Se parte de una observación *w* (palabra a corregir) y se quiere encontrar la palabra *c* que generó la palabra observada. De todas las palabras posibles del vocabulario V se quiere encontrar la palabra c tal que P(c|x) sea máxima.

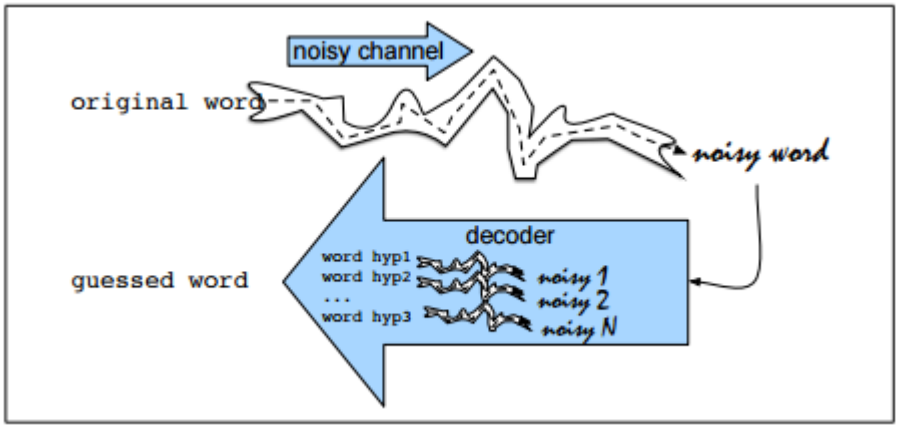

Figura 1 Modelo de canal ruidoso

La función *correction*(w) intenta elegir la corrección ortográfica más probable para w. No hay forma de saber con certeza cuál es la corrección de esta de entre una lista de posibles candidatas (por ejemplo, ¿debe "coga" ser corregido a "copa" o "cota" o "coma" o ...?) Es por ello que sugiere que usemos probabilidades de forma que estamos tratando de encontrar la corrección c, de todas las posibles correcciones candidatas, que maximiza la probabilidad de que c sea la corrección deseada, dada la palabra original w, esto es:

$$\text{argmax}_{c \in candidatas} P(c/w)$$

Basándose en el **Teorema de Bayes** esto equivale a:

$$\text{argmax}_{c \in candidatas} P(c)\, P(w/c) / P(w)$$

Como la P(w) es la misma para todas las candidatas c podemos sacarla fuera de la formula y nos quedaría:

$$\text{argmax}_{c \in candidates}\ P(c)\ P(w/c)$$

Las cuatro partes de esta expresión son:

1. Mecanismo de selección (argmax): Elegimos al candidato con la mayor probabilidad combinada.
2. Modelo Candidato (c ∈ candidatos): Esto nos indica qué correcciones candidatas, c, considerar.
3. Modelo de lenguaje (P (c)): La probabilidad de que c aparezca como una palabra de texto en español.
4. Modelo de error (P (w /c)): La probabilidad de que w se escribe en un texto cuando el autor quiere decir c. Por ejemplo, P (hyo /hoy) es relativamente más alto que P (houuiy / hoy) que sería muy bajo.

Una pregunta obvia es: ¿por qué tomar una expresión simple como P (c/w) y reemplazarlo con una expresión más compleja que implica dos modelos en lugar de uno? La respuesta es que P(c /w) ya está combinando dos factores, y es más fácil separarlos y tratarlos explícitamente.

Considere la palabra mal escrita w = "thé" y las dos correcciones candidatas c = "te" y c = "té". ¿Cuál tiene una mayor P (c/w)? Bueno, "té" parece bueno porque el único cambio es quitar la "h", que es un pequeño cambio. Por otra parte, "te" parece bueno porque "te" es una palabra muy común, aunque parece un cambio más grande, menos probable, ya que además de eliminar la "h" se ha acentuado la "e". El punto es que para estimar P (c/ w) tenemos que considerar tanto la probabilidad del candidato como la probabilidad del cambio de c a w.

5 Prototipo

Tras la conceptualización y estudio de las soluciones mencionadas en el apartado anterior. El siguiente paso fue la construcción de un prototipo de corrector ortográfico en español.

Con la construcción de dicho prototipo se perseguían principalmente dos objetivos:

- Enfrentarse a los problemas reales derivados de la construcción de un corrector ortográfico.
- Adquirir una mejor comprensión del diseño propuesto en el apartado 4, y de este modo detectar fallos en el diseño y posibles mejoras.

El prototipo está implementado en Python, y toma como base el código propuesto por Norvig. Se han realizado las modificaciones que se detallan a continuación para adaptarlo al caso específico del corrector en español y al problema propuesto en clase.

5.1 Construcción del diccionario base

5.1.1 Inicialización

Se trata de la inicialización del diccionario base. Incorpora una lista extensa de palabras pertenecientes al vocabulario en el diccionario base (WORDS).

Se han usado listas (ficheros txt) de palabras disponibles en internet con una extensión de 80383 palabras.

Se ha modificado también el código para que reciba como entrada una ruta de ficheros y no un fichero unitario. De este modo se pueden depositar los ficheros con los que se desee iniciar el diccionario.

```
def words(text):
    return re.findall(r'\w+', text.lower(), re.UNICODE)

# Inicio WORDS con las palabras de los diccionarios
# En este método no se aumentará la frecuencia de la palabra,
# si ya existe no se actualiza el numero de ocurrencias
def init_WORDS(dictionary_path):
    c_result = Counter()
    for (path, dirs, files) in walk(dictionary_path):
        for f in files:
            print('reading file {0} ...'.format(path + '/' + f))
            c_result = c_result | Counter(words(open(path + '/' + f, encoding="utf-8").read()))
    return c_result


WORDS=dict()
WORDS=init_WORDS('dictionary/es')
print('size of WORDS={0}'.format(len(WORDS)))
```

En esta fase el número de ocurrencias de la palabra no se aumenta, porque solo queremos inicializar.

5.1.2 Entrenamiento

En esta fase se utiliza el denominado modelo “bolsa de palabras” para construir el diccionario base a partir de corpus en español. Es decir, se alimenta el sistema con textos que contienen gran cantidad de palabras y sí se va actualizando el número de ocurrencias de cada una de ellas.

```
def train_WORDS(train_path):
    c_result = Counter()
    for (path, dirs, files) in walk(train_path):
        for f in files:
            print('reading file {0} ...'.format(path + '/' + f))
            ws = words(open(path + '/' + f, encoding="utf-8").read())
            c_result = c_result + Counter(ws)
    return c_result


WORDS=dict()
WORDS=init_WORDS('dictionary/es')
print('size of WORDS={0}'.format(len(WORDS)))

WORDS_train=train_WORDS('train/es')
print('size of WORDS_train={0}'.format(len(WORDS_train)))

# Elimina las palabras inexistentes en el vocabulario
# remove = [w for w in WORDS_train if exist_word_in_dict(w)==False]
# for r in remove: del WORDS_train[r]

WORDS=WORDS+WORDS_train
print('size of WORDS after training={0}'.format(len(WORDS)))
```

En ambas fases fue necesario codificar los ficheros, ya que procedían de fuentes distintas y estaban codificados de forma diferente. Para ello se usaron los siguientes comandos Unix:

```
dos2unix corpus
iconv -t UTF-8 corpus > corpusUTF8.txt
```

En la fase de entrenamiento, además fue necesario cortar los ficheros corpus de entrenamiento, ya que el tiempo de procesado de ficheros grandes era excesivamente alto y consumía más recursos de los disponibles. Para ello se utilizó el siguiente comando Unix:

```
split -l 300 corpusUTF8.txt ./line
```

5.2 Inclusión de nuevas transformaciones

En español existen caracteres adicionales a los que ya estaban definidos por Norvig en el método que realizar las transformaciones de la palabra a corregir.

Fue necesario añadir los siguientes caracteres para adaptarlo al español: ñ, á, é, í, ó, ú, ü, ï.

Por tanto, se ha modificado el método *edit1* convenientemente para que tenga en cuenta todas las posibles transformaciones de distancia <=2 de la palabra de entrada.

```
def edits1(word):
    "All edits that are one edit away from `word`."
    letters    = 'abcdefghijklmnopqrstuvwxyzñáéíóúüï'
    splits     = [(word[:i], word[i:])    for i in range(len(word) + 1)]
    deletes    = [L + R[1:]               for L, R in splits if R]
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]
    replaces   = [L + c + R[1:]           for L, R in splits if R for c in letters]
    inserts    = [L + c + R               for L, R in splits for c in letters]
    return set(deletes + transposes + replaces + inserts)
```

5.3 Integración con diccionarios online

Los corpus de entrenamiento con los que se construye el diccionario WORDS pueden contener palabras inexistentes o incorrectas, y por tanto podrían pasar a formar parte del diccionario base de nuestro corrector. El mecanismo que se ideó para evitar este problema fue la integración del diccionario ortográfico con un diccionario online.

El objetivo es el siguiente:

- Si la palabra existe en el diccionario online, se añade en el diccionario base y se actualiza el número de ocurrencias.

- Si la palabra no existe en el diccionario online no se añade al diccionario base.

Se estudiaron varias APIs de diccionarios online, entre ellas las que se listan a continuación, pero todas presentan problemas y finalmente no ha sido posible utilizar este mecanismo en la construcción del prototipo.

- **DRAE**: la API está restringida, se obtiene respuesta realizando la petición desde un navegador pero no usando una consulta automática desde un programa informático.

```
url ="http://dle.rae.es/srv/fetch?w=hola"

headers = {'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
           'Authorization': 'Basic cDY4MkpnaFMzOmFHZlVkQ2lFNDM0',
           'Accept-Encoding':'gzip, deflate', 'Content-Type':'application/x-www-form-urlencoded','Upgrade-Insecure-Requests':'1',
           'User-Agent':'Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/53

req = requests.get(url, headers=headers)
req.text
```

- **Glosbe**: Se implementó un método para comprobar la existencia de una palabra usando este diccionario. Este método discrimina con bastante exactitud cuándo la palabra pertenece al español. Sin embargo, al entrenar con corpus de gran tamaño y realizar numerosas peticiones los propietarios denegaron el acceso por superar el límite de consultas, y tuvimos que abandonar esta línea.

```
def exist_word_in_dict(w):
    result=False
    url ="https://glosbe.com/gapi/translate?from=spa&dest=spa&format=json&phrase={0}".format(w)
    print(url)
    resp = requests.get(url).json()
    if('result' in resp):
        if(resp['result'] == 'ok'):
            if('tuc' in resp):
                result=True
    return result
```

- **Wiktionary**: la respuesta que se obtiene cuando se consulta palabras en plural o algunas palabras comunes no es correcta, por ejemplo no tiene la entrada “catedráticos”. Por tanto se descartó.

- **Aonaware**: su API es bastante completa pero no está disponible en español.

Este método se llama una vez se han incorporado las palabras al diccionario y luego se realiza la limpieza de las incorrectas, de este modo conseguimos optimizar el tiempo de procesado, porque si una palabra se repite *M* veces no se realizarán M consultas a la API del diccionario.

5.4 Exportación del diccionario base

Una vez construido el diccionario base es recomendable disponer del mismo en un archivo para futuras pruebas y experimentos, y no tener que realizar el procesado y entrenamiento cada vez que se quiera usar el corrector. Para ello se ha implementado la siguiente función que exporta el contenido de WORDS a csv.

```
def export_csv():
    with open('WORDS_es.csv', 'w') as f:
        [f.write('{0},{1}\n'.format(key, value)) for key, value in WORDS.items()]
```

5.5 Lista de sugerencias

Se ha modificado el método *correction* para que devuelva una lista de sugerencias y no una única palabra. De este modo se puede conseguir interactuar con el usuario y aprender de él.

```
# Develve una lista de posibles palabras ordenadas por probabilidad
def correction(word):
    return sorted(candidates(word), key=P)
```

5.6 Principales problemas

Los principales problema a los que nos hemos enfrentado en la construcción del prototipo son los siguientes:

- Existencia de palabras incorrectas en los corpus de entrenamiento.
- Existencia de números u otras palabras que no se encuentran en el vocabulario del idioma.
- La probabilidad que se asigna a las palabras depende en gran medida del contexto del que proceda el corpus, por ejemplo si utilizo un corpus científico asignará más probabilidad a los términos científicos que a otro tipo de palabras.
- No existen listados de palabras en español con todas sus acepciones, conjugaciones, etc.
- Dificultad para verificar que la palabra que se quiere corregir realmente existe en el vocabulario del idioma.
- Rendimiento y consumo de memoria para procesar textos extensos.
- Fuentes de entrenamiento muy distintas con codificación de los ficheros diferente.

6 Propuestas de mejora

En este apartado enumeraremos algunos problemas no resueltos en la solución propuesta por Norvig, y para los que plantearemos diferentes enfoques, reflexiones y propuestas de mejora. La mayoría de las mejoras propuestas son extensibles a cualquier otro idioma pero nos centraremos en el castellano.

6.1 Niveles de revisión

Además de revisar la ortografía el corrector podría mejorarse incorporando otros niveles de revisión, como son los gramaticales, tipográficos y de estilo.

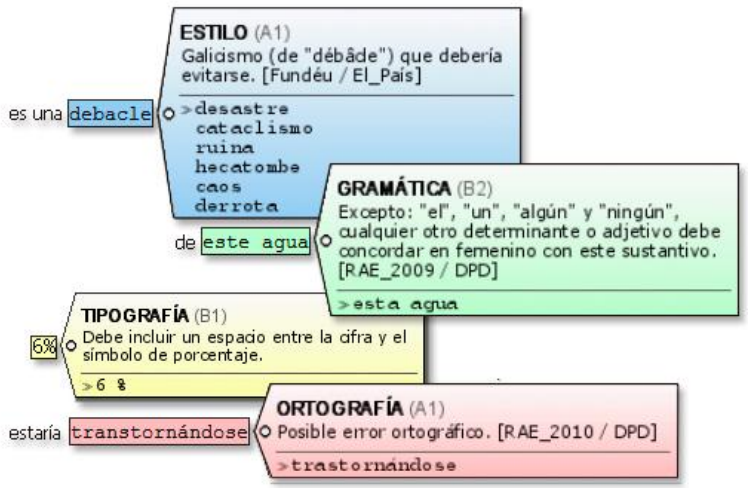

Figura 2 Distintos niveles de revisión de un texto

6.1.1 Tipografía

Algunas de las comprobaciones tipográficas que podrían incluirse al corrector son las siguientes:

- Apertura y cierre de pares de signos (comillas, paréntesis, corchetes, etc.).
- Espaciado incorrecto: dobles espacios, exigencia de espaciado o adyacencia entre signos ortográficos y palabras, etc.

- Uso de secuencias no permitidas de signos de puntuación. En el siguiente ejemplo se hace un uso redundante del punto con el cierre de interrogación:

  *¡Hasta mañana***!.**
- Uso de puntuación en las abreviaturas.
- Uso de mayúsculas y minúsculas.

6.1.2 Gramática

Sería también muy beneficioso añadir la capacidad de alertar al usuario frente a:

- Frases demasiado largas
- Discordancia entre tiempos verbales. Por ejemplo:

  La mayoría de los manifestantes **gritaba** consignas.
  La mayoría de **los manifestantes gritaban** consignas.

6.1.3 Estilo

Enriquecería el diccionario incorporando recomendaciones de uso recogidas en los manuales de dudas lingüísticas y libros de estilo más reconocidos.

Por ejemplo:

- Variantes ortográficas preferidas.
- Impropiedades léxicas.
- Propuesta de alternativas a extranjerismos y voces extranjeras.
- Detección de fenómenos que pudieran complicar la lectura como el uso abusivo de adverbios o preposiciones, frases demasiado largas, redundancias, etc.

6.1.4 Semántica

Se encarga de obtener el sentido de una oración a partir de la interacción entre las palabras que la conforman. La desambiguación del sentido de las palabras consiste en descubrir con qué sentido se está usando una palabra en un contexto dado. Por ejemplo:

Nos sentamos en el **banco** toda la tarde.
Este **banco** te da más intereses que aquel.

En general, tendremos una palabra *w*, en un contexto *c*, y se trata de descubrir, entre una serie de sentidos *{s1, ..., sk , ..., sK}*, con qué sentido se está utilizando. Por tanto, es un problema de clasificación.

Es un reto importante ya que definir qué es el significado no es una tarea sencilla, y puede dar lugar a diversas interpretaciones.

6.2 Distancia fonética y palabras homófonas

Son palabras homófonas aquellas que se pronuncian igual pero se escriben de un modo diferente y tienen significados distintos.

En el español se dan las palabras homófonas por lo que hay que tenerlas en cuenta a la hora de implementar el corrector ya que se puede dar el caso de que tengamos un error ortográfico

siendo la palabra correcta, por ejemplo, "**Hay ahí** niños" el corrector identificaría las palabras "hay" y "ahí" como correctas sin embargo estamos cometiendo dos faltas de ortografía, la frase correcta sería "**Ahí hay** niños"

En primer lugar se va a partir de un listado (PalabrasHomofonas.csv) con estos casos, este listado tiene tres columnas (h_1, h_2, h_3) ya que hay casos en los que se presentan 3 palabras homófonas (hay, ahí, ay) aunque en la mayoría de los casos son sólo dos. Por otro lado también es necesario construir una lista que contenga en una única columna las palabras homófonas (ListaHomofonas.csv). Para la detección de estas faltas ortográficas se va a proceder como sigue:

1. Comprobar si la palabra es homófona a partir de ListaHomogfonas.csv
2. En el caso de que lo sea hay que comprobar si el uso de esta es correcto o si es su homófona la correcta. Para ello se propone usar el clasificador de Naïve Bayes que habrá que aplicarlo a cada conjunto de palabras homófonas para los que se usará PalabrasHomofonas.

   Sean:

   - H: el atributo nominal que toma como valores las palabras homófonas $\{h_1,h_2,h_3\}$, normalmente será $\{h_1,h_2\}$
   - C: variable discreta que indica el conjunto al que pertenece la palabra homófona (1,…,91)
   - X1: atributo nominal que indica la palabra antecesora
   - X2: atributo nominal que indica la palabra predecesora
   - P1: Probabilidad de que ocurra H habiendo ocurrido X1 => P(H/X1)
   - P2: Probabilidad de que ocurra H habiendo ocurrido X2 => P(H/X2)

   A partir del conjunto de entrenamiento se generará esta información en un dataframe como podemos observar en la siguiente tabla:

H	**C**	**X1**	**X2**	**P1**	**P2**
baca	3	la	del	P(baca/ la)	P(baca/del)
baca	3	tiene	para	P(baca /la)	P(baca/del)
vaca	3	la	come	P(vaca/la)	P(vaca/come)
vaca	3	mi	le	P(vaca/mi)	P(vaca/le)
riza	10	se	el	P(riza/se)	P(riza/el)
risa	10	esa	es	P(risa/esa)	P(risa/es)

   El problema que resuelve el clasificador de Naive Bayes es:

$$h^* = \arg\max_{i=1,2,3} P(X1/h_i)\, P(X2/h_i)\; P(h_i)$$

   Se tratará a cada conjunto de pablaras homófonas de forma independiente, esto es que para calcular las probabilidades sólo se tendrán en cuenta las palabras $\{h_1,h_2,h_3\}$ del mismo conjunto C

   Hay que contemplar la posibilidad de que se nos presente algún caso donde $P(X1/h_i)$ o $P(X2/h_i)$ sean igual a cero debido a que en el dataframe generado con el conjunto de entrenamiento no se haya dado el caso. Es por ello que se aplicará la técnica de descuento de Laplace, esto es que se asume que todos los casos se dan al menos una vez.

En español se incluyen en este grupo también las palabras con **acento diacrítico**. El acento diacrítico es aquel que se utiliza para poder diferenciar palabras que se escriben de la misma forma pero que realmente poseen significados diferentes.

Por ejemplo:

> **Tú** tienes que estudiar para aprobar los exámenes.
> En **tu** casa tenemos planeado ver la película este fin de semana.

6.3 Aprendizaje

Será clave que el corrector ortográfico pueda aprender del usuario. Tanto para incorporar nuevos términos desconocidos (no están en el diccionario base) como para eliminar o penalizar los términos sugeridos que el usuario rechace.

Para hacer esto posible el corrector sugerirá una lista de palabras, pero no reescribirá, será el usuario quien decida, y por tanto tendrá el control de la corrección. El usuario podrá añadir nuevos términos al diccionario base. Por ejemplo términos específicos de su sector o ámbito:

- Si el usuario es informático podría tener la necesidad de incluir en su diccionario la palabra Python o py.
- Localismos: palabras usadas en regiones específicas, por ejemplo términos cordobeses como *pizco*, *perol*… si el usuario es cordobés.

Además, con cada palabra que introduzca el usuario se irá actualizando (aumentando) la frecuencia de aparición de cada palabra en el diccionario base, y de este modo el diccionario irá aprendiendo del usuario.

6.4 Asignación de pesos a las transformaciones

La idea sería asignar un peso distinto a las transformaciones que se realizan sobre la palabra original, para tener en cuenta la proximidad entre caracteres, es decir, ciertos pares de caracteres están más próximos entre sí que otros.

Por ejemplo, si tomamos la posición de las teclas de un teclado QWERTY, el carácter *p* está más cercano al *o* que al *n*.

También se podría considerar más próximos los cambios resultantes de los errores ortográficos más comunes como son intercambiar:

- b y v
- g y j
- vocal acentuada y sin acentuar
- usar n en lugar de m antes de p y b
- etc.

6.5 Cobertura léxica y tratamiento de palabras desconocidas

Con el objetivo de conseguir una mayor cobertura léxica y un mejor comportamiento cuando no exista coincidencia en el diccionario base se podrían incluir en el corrector un módulo que tenga en cuenta las construcciones más comunes de palabras a partir de otras o de una raíz.

6.5.1 Conjugación verbal

Se podrían usar técnicas de etiquetado gramatical (**POS tagging**) para proporcionar al corrector la capacidad de identificar en los textos que introduzca el usuario formas verbales. El corrector clasificará dentro de estas tres categorías

- Verbos regulares e irregulares
- Modelos de conjugaciones: 1ª conjugación, 2ª conjugación, etc.
- Tiempos verbales: presente, pasado, futuro… En este punto será necesario contar con el contexto del texto, partículas temporales, como *ayer*, *mañana*, etc.

Y aplicará las reglas más adecuadas para conjugar el verbo y obtener palabras nuevas a partir de esta.

6.5.2 Elementos compositivos y palabras derivadas (sufijos y prefijos)

Se podría incluir un módulo que contenga un listado con los principales prefijos y sufijos, y que tomando como base la palabra origen realice transformaciones de ésta consistentes en añadir los prefijos y sufijos y calcular su probabilidad para determinar si se ofrece al usuario como alternativa posible o no.

6.5.3 Abreviaturas

Otra posible mejora sería enriquecer el vocabulario base con las abreviaturas aceptadas por el idioma.

Se partirá de una lista donde se recogen las abreviaturas convencionales más usuales en español. Se trata de una lista necesariamente incompleta, ya que cualquier usuario de la lengua puede crear cuantas abreviaturas considere oportunas, siempre que lo haga de acuerdo con las reglas de formación de este tipo de abreviaciones

Por ejemplo

A.	Alteza
(a)	Alias
A/A	a la atención

6.5.4 Símbolos alfabetizables

Se podría incluir de igual modo en el diccionario base una lista donde se recogen los símbolos alfabetizables más usuales, casi todos ellos referidos a las unidades de medida, los elementos de la tabla periódica, los puntos cardinales y las monedas oficiales de todos los países europeos y americanos, así como de Filipinas y Guinea Ecuatorial.

Por ejemplo:

A	Amperio
Ag	Plata
Al	Aluminio

6.6 Actualización

El lenguaje está vivo, continuamente aparecen nuevas palabras, reglas y desaparecen o dejan de usarse muchas otras. Es importante que el diccionario se mantenga actualizado y que continuamente incorpore los neologismos, así como los últimos cambios registrados por las autoridades lingüísticas de cada idioma.

6.7 Detección de idioma

Integrar con el detector de lenguaje del primer ejercicio, para que automáticamente detecte el idioma y elija el diccionario adecuado.

6.8 Inteligencia colectiva

Se podría también utilizar como fuente para el aprendizaje, aplicaciones como el popular juego *apalabrados*, en la que diariamente se construye gran número de palabras en todos los idiomas en muy distintos ámbitos y sectores.

7 Conclusiones

Una de las piezas fundamentales del corrector propuesto es el diccionario base que usará como patrón de corrección y para sugerir la mejor lista de palabras candidatas. En la construcción de dicho diccionario es esencial el entrenamiento con textos representativos en el idioma correspondiente. Para obtener buenos resultados hay que tener en cuenta lo siguiente:

- **Contexto**: tanto el entorno lingüístico que acompaña a una palaba, como el lugar, la audiencia, el entorno, el momento, influyen en el la elección del mejor candidato.
- Las **fuentes** utilizadas para el **entrenamiento**. El origen del que proviene el texto influye en gran medida en el proceso de construcción del corrector (diccionario base), puesto que hará más probable la aparición de un conjunto de palabras.
- **Rigor lingüístico**: las fuentes de entrenamiento pueden contener términos erróneos. Es necesario una verificación con un diccionario riguroso y validado por los organismos competentes para cada idioma.
- Necesidad de **pre-procesado** de las fuentes.
- **Tiempo** y **recursos** de **procesado**: es necesario disponer de máquinas potentes que minimicen el tiempo necesario para procesar las fuentes para entrenamiento.

El otro punto clave es la capacidad de aprendizaje del corrector y de adaptarse al usuario que lo está utilizando. Todo esto unido a los mecanismos propuestos como mejoras (apartado 6) podemos conseguir una tasa de acierto alta.

8 Referencias

[1] http://norvig.com/spell-correct.html

[2] Mays, Eric, Fred J. Damerau and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 23(5), 517–522

[3] http://liceu.uab.cat/~joaquim/language_technology/NLP/PLN_aplicaciones.html

[4] http://www.mystilus.com

[5] https://es.wikipedia.org/wiki/Acento_diacr%C3%ADtico

[6] http://www.rae.es/diccionario-panhispanico-de-dudas/apendices/simbolos-alfabetizables

[7] http://www.rae.es/diccionario-panhispanico-de-dudas/apendices/abreviaturas

[8] http://lema.rae.es/dpd/?key=abreviatura

[9] http://www.rae.es/sites/default/files/Elementos_compositivos_prefijos_y_sufijos_del_espanol_Esencial.pdf

[10] http://giusseppe.net/blog/archivo/2015/10/29/diccionario-de-la-rae-en-modo-texto-plano/

[11] https://raw.githubusercontent.com/javierarce/palabras/master/listado-general.txt