UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

FACULTAD DE INGENIERÍA

Construcción de herramientas de soporte para corrección en enseñanza de inglés

2021

Estudiantes:

Romina Brown Latierro

Santiago Páez Castro

Docentes:

Luis Chiruzzo

Aiala Rosá

Resumen

En estos últimos años se ha buscado la universalización de la enseñanza de inglés en todas las escuelas públicas del país. No solo en escuelas urbanas, sino en escuelas rurales que debido a su ubicación poseen escasos recursos o herramientas para llevar a cabo esta tarea. La baja conectividad a Internet y la escasez de docentes de inglés son algunas de estas limitantes. Esto genera que sea difícil para los niños alcanzar los niveles básicos de inglés al terminar la educación primaria.

Este proyecto es parte de una serie de proyectos que pertencen a la colaboración entre el grupo de PLN de la Facultad de Ingeniería - UdelaR y el Programa de Políticas Linguísticas (PPL) - ANEP. Los proyectos buscan generar herramientas y recursos que den soporte tanto a estudiantes como a docentes a la hora de llevar adelante las diferentes actividades de la enseñanza de inglés teniendo en cuenta las diferentes dificultades y limitantes presentes. En este marco es que se genera este proyecto, al buscar una solución para ayudar a los docentes en la corrección de ejercicios donde los estudiantes escriben textos cortos y "libres" en respuesta a ejercicios donde pueden describir una imagen, un video o responder preguntas referentes a una porción de texto.

A partir de un muestreo de textos proporcionados por "Ceibal en Inglés" y realizados por niños escolares donde debían describir una imagen, se determinaron los errores cometidos con mayor frecuencia por parte de los estudiantes para construir un corrector de oraciones declarativas que detecte y marque en el texto estos tipos de errores. Para construir el corrector se investigaron las técnicas y herramientas utilizadas dentro del área de Corrección de Errores Gramaticales, área perteneciente al Procesamiento del Lenguaje Natural y para cada tipo de error seleccionado se exploraron diferentes heurísticas de detección.

Una vez finalizada la implementación del corrector y dado que los textos poseen un puntaje asignado por un profesor, se entrena un modelo de aprendizaje automático utilizando la información proporcionada por el corrector sobre los tipos de error encontrados para poder así clasificar nuevos textos y asignarles un puntaje de forma automática. Los resultados obtenidos se comparan con un trabajo realizado previamente sobre el mismo conjunto de textos el cual tiene el mismo objetivo pero utiliza técnicas diferentes.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

La enseñanza de inglés en las escuelas públicas de nuestro país ha ido ganando importancia con el correr de los años y el desarrollo de la globalización. Vivimos en un mundo donde es cada vez más relevante poder interpretar y comunicarse a través de esta lengua extranjera. El idioma inglés no solo es trascendente a nivel académico y científico sino que es muy importante para la comunicación intercultural.

Con el fin de atender a esta necesidad, la Administración Nacional de Educación Pública (ANEP) estableció en su proyecto "Uruguay Plurilingüe 2030" que todos los alumnos que egresen de bachillerato de la Enseñanza Media pública deberán alcanzar un nivel B2 en dicho año. Para alcanzar ese objetivo es importante que los alumnos reciban formación apropiada en este idioma desde la escuela Primaria, de forma de alcanzar dicho nivel al final del bachillerato. Con este horizonte claro, se universalizó la enseñanza del inglés a partir de 4^{to} año de educación primaria. Desde 2014 dicha universalización se alcanzó en las escuelas urbanas gracias a los programas de presencialidad de segundas lenguas y de educación remota por parte del programa "Ceibal en Inglés" (CEI)[1].

La misión de CEI es «Promover el aprendizaje de inglés como lengua extranjera en todos los niveles de educación pública para mejorar su calidad, fomentar la interculturalidad y favorecer mayor inclusión y crecimiento personal utilizando la tecnología para lograr este propósito». A través de equipos de videoconferencia instalados por Plan Ceibal en los centros educativos, los estudiantes trabajan una vez por semana con un profesor remoto y se complementa con actividades conducidas por el docente del aula en otras instancias semanales. [2]

Esta solución resulta inapropiada para las escuelas rurales del Uruguay que presentan más de 17.000 alumnos matriculados en todo el país según los datos obtenidos en 2019[3]. La mayoría de las escuelas rurales presentan problemas de accesibilidad para que el docente de segundas lenguas pueda acudir al centro. Por otra parte, el acceso a Internet es limitado lo que impide que la enseñanza por videoconferencia sea la forma de atender a dicha escuela. Sumado a esto, es necesario considerar que la mayoría de las escuelas rurales son multigrado. Allí existe un maestro por escuela encargado de dictar todos los contenidos, incluso de distintos años. Ello plantea un desafío ya que los cursos no se encaran como

[1] `https://www.anep.edu.uy/sites/default/files/ISL.pdf`

[2] `https://ingles.ceibal.edu.uy/sobre-cei`

[3] Extraído del boletín estadístico anual de educación pública a 2019 accesible en `https://www.dgeip.edu.uy/divisiones/planeamiento-educativo/`

cursos de 4^{to}, 5^{to} y 6^{to} sino como primer año, segundo año y tercer año de exposición a la lengua inglesa, comenzando desde el nivel de Educación Inicial. Para estas escuelas rurales existe el programa "Inglés sin límites": una propuesta que no depende de la presencia de un docente de inglés y que tampoco necesita que exista conectividad a Internet para su funcionamiento.

En este proyecto se propone la aplicación de técnicas de Procesamiento de Lenguaje Natural (PLN) para la construcción de herramientas que den soporte a la enseñanza de inglés a alumnos de escuelas rurales en diferentes departamentos del Uruguay. Existen otros proyectos de grado dentro de esta área, donde se crearon herramientas que generan ejercicios y sus correcciones de forma automática permitiendo al docente editar errores que se hayan generado por la aplicación de herramientas de PLN. Cada uno de estos proyectos busca contribuir facilitando el acceso a recursos para la enseñanza del inglés. Buscando contribuir en este desarrollo y con apoyo del Programa de Políticas Linguísticas (PPL) - ANEP, este trabajo consiste en la realización de un corrector de errores en textos cortos escritos en un nivel inicial de inglés, donde se detectan los errores cometidos por el estudiante (siendo marcados en el texto) y se corrigen.

Posteriormente, gracias a la contribución de "Ceibal en Inglés" de un corpus con 65528 textos, se experimentó con el corrector de errores para realizar un clasificador de textos.

1.1. Objetivo general

Teniendo en cuenta las dificultades que enfrenta el maestro rural al enseñar inglés, este proyecto se propone desarrollar métodos para la corrección automática de textos cortos escritos por estudiantes de nivel inicial de inglés. El objetivo principal es dar soporte al maestro ubicando en el texto los errores detectados por la herramienta. Contando con esta información proporcionada por el detector, podrá finalizar la corrección indicándole al niño sus errores.

1.2. Objetivos específicos

A continuación se lista los objetivos específicos que se persiguieron para lograr el objetivo general.

- Conocer herramientas de análisis lingüístico usuales en el área PLN.
- Estudiar trabajos vinculados a la aplicación de PLN a la enseñanza de lenguas, particularmente los trabajos desarrollados por estudiantes de proyecto de grado en el año 2018.
- Estudiar trabajos relacionados a la corrección automática de trabajos escritos por estudiantes.
- Implementar y evaluar distintas técnicas para la corrección de errores en textos de inglés.
- Implementar y evaluar un corrector automático de errores basado en heurísticas.

- Construir y evaluar la calidad de un clasificador de textos a partir de las heurísticas obtenidas por el detector de errores.

1.3. ¿Por qué elegimos el proyecto?

Una de las principales razones que nos impulsaron a elegir esta propuesta, es su relación con la educación. En particular, porque se trata de una herramienta que podrá contribuir en el trabajo que realizan maestros de la zona rural de nuestro país. Creemos que es cada vez más importante desarrollar recursos tecnológicos en el ámbito educativo con el fin de facilitar la transmisión de conocimientos a los niños.

Por otra parte, el interés por el Procesamiento de Lenguaje Natural se despertó en nosotros al cursar algunas asignaturas referidas al área. Con la idea de investigar y seguir aprendiendo sobre la temática, decidimos embarcarnos en el desarrollo de este proyecto cuya área principal es PLN.

1.4. Organización del documento

A continuación se describe la estructura general de los siguientes capítulos.

En el capítulo 2 llamado Marco Teórico, se realiza una introducción al Procesamiento de Lenguje Natural incluyendo técnicas utilizadas en el proyecto. También se describe el Aprendizaje Automático y una de sus aplicaciones: resolver problemas de clasificación. Contiene también un resumen de distintos trabajos referidos a la enseñanza de diferentes lenguas mediantes el uso del Procesamiento de Lenguaje Natural y que sirvieron como puntapié para el desarrollo del presente trabajo.

En el capítulo 3, Descripción y análisis de los datos, se introduce el conjunto de datos que se utilizó para el proyecto. Se especifica y analiza su estructura y se describen los muestreos utilizados para iniciar el trabajo.

En el capítulo 4, Corrector, se presenta en detalle el corrector implementado describiendo sus etapas y cada uno de los errores detectados. También se incluyen ejemplos del marcado de textos y se analizan los resultados obtenidos.

En el capítulo 5, Clasificador, se presenta un trabajo previo realizado sobre el mismo corpus para luego describir el trabajo de clasificación que se realizó en base a los errores detectados por la herramienta desarrollada. A continuación en el mismo capítulo se procede a comparar los clasificadores y sus resultados.

En el capítulo 6, Conclusiones y trabajo a futuro, se detallan las conclusiones obtenidas y aspectos en los que se podría continuar trabajando para mejorar lo ya hecho.

Por último, se incluye la Bibliografía y Anexos.

Capítulo 2

Marco teórico

2.1. Procesamiento de Lenguaje Natural

El Procesamiento del Lenguaje Natural (PLN) es el área de estudio dentro de la computación que se encarga de investigar y crear herramientas que faciliten la utilización y comprensión del lenguaje natural, ya sea en forma de texto o de habla por parte de sistemas informáticos. Posee puntos de contacto con otras disciplinas, como la Lingüística, la Psicología y la Inteligencia Artificial (IA).

El PLN precisa conocimientos sobre distintas ramas de la Lingüística:

- **Fonética:** Estudio de la naturaleza física de los sonidos del discurso humano.
- **Fonología:** Estudio de cómo funcionan los sonidos en la lengua, incluyendo las sílabas, la entonación, la acentuación, etc.
- **Morfología:** Estudio de la estructura interna de las palabras.
- **Sintaxis:** Estudio de la estructuración (orden y agrupamiento) de las palabras en unidades mayores.
- **Semántica:** Estudio del significado.
- **Pragmática:** Estudio de cómo el lenguaje se utiliza para cumplir objetivos.
- **Discurso:** Estudio de las unidades mayores a la oración.

El objetivo de este campo es lograr que sistemas informáticos ejecuten tareas útiles relacionadas al lenguaje natural, tareas como habilitar la comunicación humano-máquina, mejorar la comunicación humano-humano, o simplemente realizar procesamiento útil de texto o habla [20].

Dentro de estas tareas podemos encontrar:

- **Chatbots o asistentes virtuales**
- **Detección de spam**

- **Traducción automática**
- **Análisis de sentimiento en redes sociales**
- **Resumen de documentos**
- **Corrección de errores gramaticales**

2.1.1. Problema de ambigüedad

Uno de los principales problemas al cual nos enfrentaremos a la hora de explorar una gran parte de las tareas de PLN es el problema de la ambigüedad en el lenguaje.

El lenguaje natural es ambiguo. Se encuentra en constante cambio y evolución. Las personas son buenas a la hora de producir y entender el lenguaje, y son capaces de expresar, percibir, e interpretar significados muy elaborados y matizados. Mientras los humanos somos buenos en esta tarea, no logramos ser tan buenos a la hora de describir o entender de manera formal las reglas que gobiernan al lenguaje [14].

Si estamos trabajando con el habla en el idioma inglés, tenemos por ejemplo el caso de la palabra "lead", la cual se pronuncia de una manera en la oración "This paint has **lead**" y de otra en "Take the **lead** please". Un sistema TTS (*Text-to-Speech*) necesitará utilizar el contexto para decidir cual es la pronunciación correcta. Por otra parte, un sistema ASR (*Automatic Speech Recognition*) encontrará ambigüedades en las palabras homófonas. Estos tipos de sistemas deberán decidir si el orador dijo "The **sail** of a boat." o "The **sale** of a boat." ya que "sail" y "sale" se pronuncian de la misma manera.

En cuanto al trabajo con texto en inglés, en la oración "The soldiers like the **port**", la ambigüedad (léxica) se encuentra en la palabra "port", la cual puede referir tanto al puerto como al vino. En el estudio realizado en [3], se estima como una cota inferior que el 32 % de las palabras utilizadas en el idioma inglés pueden ser léxicamente ambiguas.

Este problema se ve en general en todas las etapas del PLN, como veremos en ejemplos más adelante.

2.1.2. Pipeline de PLN

Generalmente, a la hora de resolver un problema de PLN que involucre texto, un primer paso es procesarlo mediante una serie de tareas consecutivas denominadas *pipeline* de PLN. Este *pipeline* tiene como objetivo obtener información lingüística sobre el texto, la cual podremos utilizar luego para tomar decisiones específicas que nos ayuden en la resolución de nuestro problema en particular.

Cada problema a ser resuelto determinará el *pipeline* de forma específica, pudiendo tener componentes distintos dependiendo del problema a resolver. Aun así, existen tareas generales que son necesarias a la hora de resolver una gran parte de los problemas y utilizaremos, entonces, un ejemplo de *pipeline* para describir estos componentes. Este ejemplo tiene los componentes generalmente ofrecidos por las distintas librerías de PLN en las cuales nos apoyamos para obtener información lingüística sobre el texto.

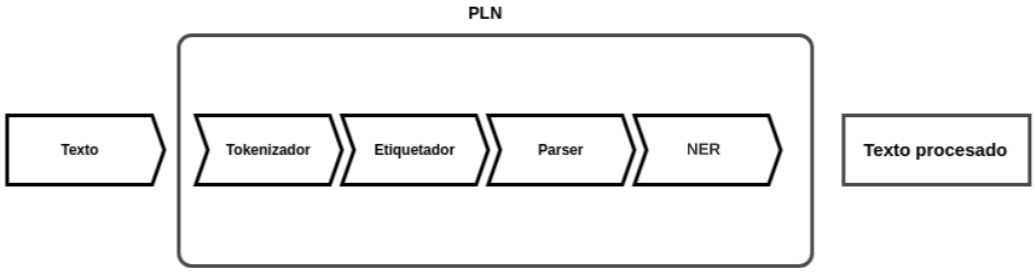


Figura 2.1: Ejemplo de *pipeline* ofrecida por librerías de PLN.

Los distintos componentes que realizan las tareas que forman parte del *pipeline* en la figura 2.1 son: Tokenizador, Etiquetador morfológico, Parser, y NER. Cada uno de estos componentes puede ser implementado utilizando diferentes métodos y técnicas, y las tareas de PLN asociadas a estos componentes son áreas activas de investigación y desarrollo.

La librería *spacy*[1], por ejemplo, ofrece un *pipeline* donde podemos encontrar también otros componentes (como *word embeddings* y *lemmatizer*) y donde podemos indicarle a la librería si queremos habilitar o deshabilitar alguno de estos componentes al utilizarla.

2.1.2.1. Tokenizador

La tokenización se refiere a la tarea de fragmentar un texto en palabras u oraciones, a las cuales llamamos *tokens*. En el caso en que estemos buscando realizar una tokenización por palabras, en el idioma inglés, podemos realizar esta separación utilizando el espacio en blanco que existe entre palabras. Esto no ocurre en otros lenguajes como el chino o japonés.[19]

Si bien esta primera separación cubre la mayor parte de los casos, es necesario tener en cuenta algunos otros casos a la hora de obtener los *tokens*. Es necesario separar los símbolos de puntuación de las palabras, y tener en cuenta que existen *tokens* que poseen símbolos de puntuación internos como por ejemplo “a.m.” y “www.fing.edu.uy”.

Los números, fechas y el uso del apóstrofo (posesivo o de contracción) también deben observarse con cuidado. El número “21.3” debería ser interpretado con un único *token* y no con dos *tokens* “21” y “3”. En cuanto al uso del apóstrofo, al encontrar “I’m” se deberían generar los *tokens* “I” y “am”.

Si buscamos realizar la separación en oraciones, en el idioma inglés, los símbolos de puntuación se encargan de finalizar oraciones. Al igual que en el caso de la tokenización en palabras, el uso del punto por ejemplo puede ser ambiguo. Puede determinar el fin de una oración o puede determinar una abreviación como en el caso de “Mr.” o “Mrs.”.[19]

2.1.2.2. Etiquetador morfológico

El proceso de etiquetado, o *Part-of-Speech tagging* (*POS tagging*) se refiere a la tarea de asignar etiquetas a las palabras de un corpus representando su categoría gramatical (sustantivo, verbo, preposición, determinante, etc) e información morfológica.

[1] `https://spacy.io/`

Token	tag Brown	tag Penn	tag C5	tag spacy	tag UD
She	PPS	PRP	PNP	PRP	PRON
is	BBB	VBZ	VBZ	VBZ	AUX
14	CD	CD	CRD	CD	NUM
years	NNS	NNS	NN2	NNS	NOUN
old	JJ	JJ	SENT	JJ	ADJ
.	.	.	PUN	.	PUNCT

Tabla 2.1: Resultado de etiquetar utilizando distintos *tagsets* la oración "She is 14 years old."

Los métodos de etiquetado (entre otras tantas tareas de PLN) pueden dividirse generalmente en dos categorías: métodos basados en reglas o métodos estocásticos. Los métodos basados en reglas buscan generar una colección de reglas para en este caso determinar la clase morfológica de una palabra mientras que los métodos estocásticos se basan en entrenar modelos utilizando un corpus anotado para determinar la probabilidad de que una palabra pertenezca a cierta clase morfológica dado su contexto.

Existen también métodos híbridos como el utilizado por el etiquetador Brill [2], el cual utiliza reglas aprendidas del corpus de entrenamiento mediante aprendizaje automático.

Los corpus anotados definen sus propios conjuntos de categorías gramaticales o *tagsets*. Hay un pequeño número de *tagsets* populares para el idioma inglés [20]. El *tagset* utilizado en el corpus Brown [10], consta de 87 categorías gramaticales, y el corpus anotado posee 1 millón de palabras correspondientes a una muestra de 500 textos escritos. Otros *tagsets* populares son el Penn Treebank [22] y C5 (CLAWS) [11] con 45 y 61 categorías gramaticales respectivamente.

La diferencia en la cantidad de categorías gramaticales determina el nivel de granularidad. Se puede etiquetar a los sustantivos como una categoría gramatical o se puede elegir ser más específico eligiendo etiquetar si el sustantivo es un nombre propio o un sustantivo común. Podemos a su vez, etiquetar los sustantivos comunes como contables o no contables, singulares o plurales.

Por ejemplo, para la oración "She is 14 years old." las etiquetas asignadas utilizando los distintos *tagsets* se muestran en la tabla 2.1. Se agregó a los *tagsets* mencionados, el utilizado por la librería *spacy*[2], la cual utiliza el *tagset* propuesto en UD (*Universal Dependencies*)[3] un proyecto *open-source* que posee un bajo nivel de granularidad ya que busca crear anotaciones que funcionen a través de múltiples lenguajes. Spacy ofrece también etiquetar utilizando OntoNotes 5.0[4] basado en el Penn Treebank *tagset* el cual provee un mayor nivel de granularidad.

Dos problemas comunes a la hora de etiquetar texto son: encontrar palabras desconocidas (que no pertenecen al conjunto de entrenamiento o no son consideradas por las reglas) y problemas de ambigüedad.

El primer problema sucede porque los lenguajes están "vivos" en el sentido en el que evolucionan y cambian, agregando nuevas palabras o modificando palabras existentes para

[2] `https://spacy.io/`

[3] `https://universaldependencies.org/` - Último acceso: Enero 2021

[4] `https://catalog.ldc.upenn.edu/LDC2013T19` - Último acceso: Enero 2021

reflejar nuevos conceptos. Las categorías gramaticales reflejan este fenómeno con la idea de clase abierta o clase cerrada. Una clase abierta es aquella que admite nuevas palabras, y en el caso del inglés tenemos a los sustantivos y a los verbos como ejemplo de categorías gramaticales abiertas. Una clase cerrada es aquella donde es relativamente raro que se introduzcan nuevas palabras o que el significado de las mismas cambie. En el caso del inglés tenemos a los determinantes y a las preposiciones como ejemplos de dichas clases.

Las estrategias utilizadas al encontrar una nueva palabra generalmente son dos: asumir que una palabra desconocida tiene una frecuencia similar a las palabras que aparecen una sola vez en el corpus e inferir una etiqueta de esta manera, o utilizar información morfológica de la palabra desconocida para tomar una decisión. Por ejemplo, si la palabra comienza en mayúscula, puede ser un nombre propio, si finaliza en "-s", puede ser un sustantivo plural.

El otro problema a resolver cuando se etiquetan palabras se refiere a su ambigüedad. Una palabra se cataloga como ambigua cuando puede ser utilizada de distintas maneras, por lo cual debe ser etiquetada de distintas maneras. Por ejemplo la palabra "book" puede ser un verbo al ser utilizada en: "I'm going to book that flight." o un sustantivo en "Have you read that book?". La resolución de estas ambigüedades se realiza utilizando los métodos mencionados anteriormente (por reglas y estocásticos) determinando la etiqueta dado el contexto.

2.1.2.3. Parser

El *parser* es el encargado de reconocer la estructura sintáctica de una oración. Lo realiza construyendo un árbol que indica cómo se relacionan las distintas palabras. Existen distintos tipos de *parsers* los cuales se diferencian en los algoritmos y las gramáticas que utilizan a la hora de construir el árbol de *parsing* resultado.

Una gramática busca modelar mediante reglas las distintas estructuras gramaticales del lenguaje. Generalmente se utilizan *treebanks* o corpus anotados de forma semi-automática (árboles generados por herramientas automáticas, luego corregidos por lingüistas) para derivar las reglas de la gramática.

Las gramáticas basadas en constituyentes buscan identificar y relacionar las unidades llamadas constituyentes. Estas unidades pueden ser conformadas por una palabra o un conjunto de palabras, pero se comportan como una unidad en el contexto de la oración. Un ejemplo de este tipo de unidades puede ser "she" y "a well-weathered three-story structure" siendo estos dos sintagmas nominales (NP: *Noun Phrase*). A su vez otros tipos de constituyentes pueden ser los sintagmas verbales (VP: *Verb Phrase*), o sintagmas preposicionales (PP: *Prepositional Phrase*), entre otros.

Las gramáticas basadas en dependencias representan la estructura de la oración solamente utilizando las palabras y un conjunto de relaciones gramaticales expresado mediante arcos dirigidos y etiquetados. Estos arcos etiquetados indican la relación de dependencia entre las palabras, por ejemplo *nsubj* representando al sujeto y *dobj* al objeto directo del predicado.

En la figura 2.2 tenemos un ejemplo de un árbol de dependencia con los arcos etiquetados mientras que en la figura 2.3 podemos comparar la diferencia estructural entre un árbol generado utilizando una gramática de dependencia y una gramática de

constituyentes sobre la misma oración.

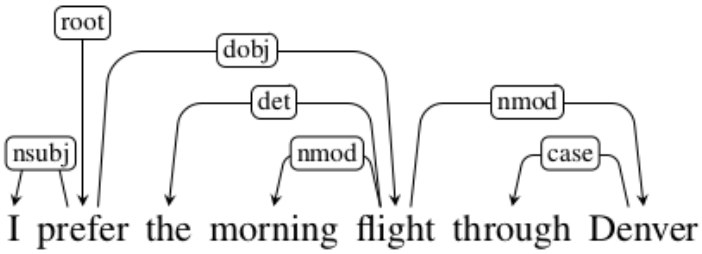


Figura 2.2: Árbol de *parsing* de dependencia sobre la oración "I prefer the morning flight through Denver" [20]

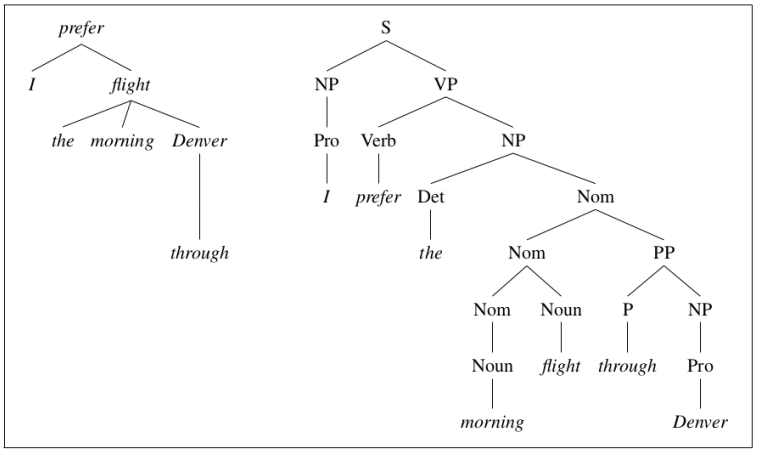


Figura 2.3: Estructuras de árboles de *parsing* al utilizar una gramática de dependencia (izquierda) y una gramática de constituyentes (derecha) sobre la oración "I prefer the morning flight through Denver" [20]

Un *parser* de constituyentes generará estructuras (árboles de parseo) que siguen las reglas establecidas por la gramática mientras que un *parser* de dependencia generará árboles que denoten las relaciones de dependencia entre las palabras mediante arcos etiquetados. En cuanto a los *treebanks* utilizados, el *Penn Treebank Project* [22] posee un *treebank* anotado con 4.5 millones de palabras etiquetadas (*POS tagged*) y la mitad de éstas anotadas con su estructura sintáctica (basada en constituyentes). Mientras que el proyecto UD[5] (*Universal Dependencies*) posee alrededor de 200 *treebanks* cubriendo más de 100 idiomas con anotaciones utilizando una gramática de dependencias.

La ambigüedad es el problema más serio que deben enfrentar los *parsers* sintácticos. En este contexto, la ambigüedad se presenta en la forma de ambigüedad estructural y se manifiesta en oraciones a las cuales se les puede asignar múltiples árboles de *parsing* válidos. Se pueden reconocer dos tipos de ambigüedades estructurales: *attachment ambiguity* y *coordination ambiguity* [20].

La primera ocurre cuando un constituyente puede ser agregado al árbol de *parsing* en múltiples lugares, por ejemplo en la oración "We saw the Eiffel Tower flying to Paris." el constituyente "flying to Paris" puede considerarse parte de la unidad cuyo sujeto es

[5] `https://universaldependencies.org` - Último acceso: Enero 2021

"the Eiffel Tower" o puede considerarse un adjunto el cual modifica la frase verbal que contiene a "saw". Este tipo de ambigüedades es fácil de resolver para una persona y las oraciones donde ocurren pueden ser utilizadas como bromas, dado que si bien existen varias interpretaciones gramaticalmente correctas, algunas de ellas no tienen sentido semántico (en el ejemplo anterior, es fácil entender/deducir que la Torre Eiffel no vuela).

La ambigüedad de coordinación suele ocurrir en oraciones donde existen diferentes conjuntos de frases las cuales pueden unirse mediante una conjunción como por ejemplo "and". En la oración "Old men and women can be happy.", se puede interpretar como ["old men", "women"] haciendo referencia a hombres mayores y mujeres (sin restricción de edad) o ["old" ["men and women"]] haciendo referencia a hombres y mujeres mayores.

El proceso mediante el cual los *parsers* manejan estos tipos de ambigüedades se llama desambiguación sintáctica e involucra conocimiento estadístico, semántico y pragmático [20]. En caso de no poder realizar la desambiguación, la opción disponible es la de retornar los múltiples árboles de *parsing* válidos para la oración.

2.1.2.4. Reconocimiento de Entidades con Nombre

La tarea de Reconocimiento de Entidades con Nombre o NER por su sigla en inglés (*Named Entity Recognition*) busca identificar segmentos del texto que correspondan a nombres propios y asignarles una etiqueta que identifique el tipo de entidad al que refieren. En general se utilizan cuatro etiquetas: **PER** (persona), **LOC** (locación), **ORG** (organización) y **GPE** (entidad geo-política) aunque se suele expandir el conjunto de etiquetas para agregar expresiones temporales como fechas y horas, y expresiones numéricas como precios [19].

En la figura 2.4 podemos ver un ejemplo de una oración etiquetada en la que cuatro entidades fueron identificadas. Una entidad persona ("John"), una entidad locación ("New York"), una entidad organización ("ACME Corp.") y una entidad temporal ("2005").

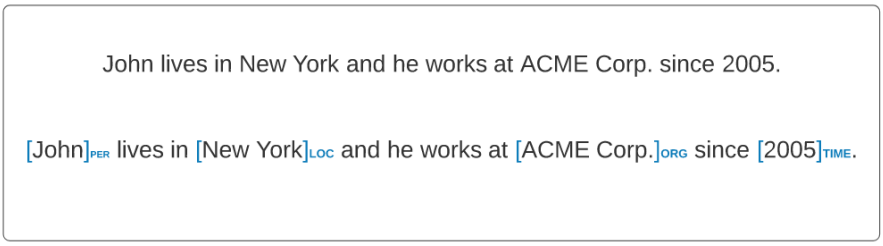


Figura 2.4: Resultado al etiquetar la oración utilizando reconocimiento de entidades con nombre.

El problema de etiquetar entidades con nombre es similar al problema de *POS tagging*, en cuanto que aquí también buscamos asignar etiquetas a las palabras. Una diferencia entre ellos en este sentido es que en *POS tagging* buscamos asignar una única etiqueta a cada palabra mientras que en *NER*, buscamos asignar una única etiqueta a segmentos de texto. Una estrategia comúnmente utilizada a la hora de etiquetar secuencias donde debemos identificar segmentos de la secuencia es utilizar *BIO tagging* [19].

Para realizar el etiquetado BIO, etiquetamos utilizando la etiqueta **B** al encontrar una palabra que comienza un segmento de interés (un segmento que representa a una entidad

con nombre en este caso). Palabras dentro de un segmento de interés serán etiquetadas con **I**, mientras que palabras que no pertenezcan a un segmento de interés serán etiquetadas con **O**. Existe una única etiqueta **O**, y una etiqueta **B** e **I** por cada tipo de entidad. Por lo que se cuenta con un total de $2n + 1$ etiquetas, donde n es la cantidad de tipos diferentes de entidades que buscamos identificar. La figura 2.5 muestra el ejemplo de una oración etiquetada utilizando *NER* y la misma oración etiquetada utilizando *BIO*.

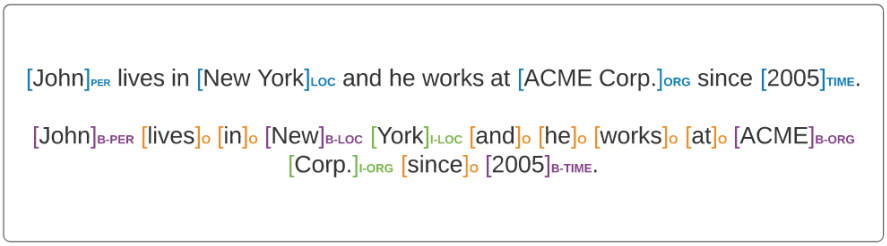


Figura 2.5: Etiquetado *NER* y etiquetado *BIO* de una oración.

Ambas formas de etiquetar encuentran los mismos segmentos de interés, pero al utilizar el etiquetado *BIO* obtenemos una etiqueta por palabra lo que nos permite utilizar técnicas para etiquetar secuencias (*sequence labeling*) utilizadas también en *POS tagging* y en el análisis sintáctico superficial (*shallow parsing*).

Si bien existen métodos basados solamente en reglas y métodos basados solamente en aprendizaje automático para resolver este tipo de problemas, las soluciones comerciales suelen utilizar una combinación de ambos donde se le realizan varias pasadas al texto utilizando reglas y luego se utilizan técnicas de aprendizaje automático agregando la información obtenida al aplicar las reglas junto con otros *features* para clasificar una palabra y obtener su etiqueta. [19]

Los sistemas de reconocimiento de entidades con nombre se enfrentan a dos tipos de problemas de ambigüedad. El primero se presenta cuando un mismo nombre puede referirse a dos entidades distintas del mismo tipo. Un ejemplo de esto es *JFK* la cual es la abreviación del nombre John Fitzgeral Kennedy, ex-presidente de los Estados Unidos, pero también es el nombre de su hijo.

El ex-presidente norteamericano es también un buen ejemplo para el segundo tipo de ambigüedad, el cual sucede cuando una misma palabra puede representar a más de un tipo de entidad. La abreviación *JFK* puede hacer mención no solo a la persona, sino también al aeropuerto de Nueva York, o a varios puentes y liceos a lo largo de todo el país norteamericano.

2.2. Corrección de errores gramaticales (GEC)

La corrección de errores gramaticales (*GEC: Grammatical Error Correction*) es una tarea dentro del Procesamiento del Lenguaje Natural que consiste en corregir los diferentes tipos de errores encontrados en un texto. Estos errores pueden ser errores ortográficos, de puntuación, gramaticales y de elección de palabras entre otros.

El interés en el estudio de la corrección automática de errores surge, por un lado, ante la necesidad de desarrollar métodos capaces de detectar errores en la etapa de entrada

manual de datos en sistemas informáticos [33]. Estos métodos fueron luego incorporados, por ejemplo, en componentes de sistemas de producción de texto que se encargan de detectar errores de ortografía o estilo.

Por otro lado, existe el interés en generar herramientas de soporte a profesores y estudiantes de idiomas. Estos sistemas llamados *CALI - Computer Assisted Language Instruction* o *CALL - Computer Assisted Language Learning* buscan detectar, corregir y en algunos casos dar sugerencias acerca de los errores encontrados. El sistema *Scripsi* [6], cuyo artículo fue publicado en el año 1990, utiliza un parser junto con una base de datos de reglas gramaticales para detectar errores. Provee una devolución a modo de sugerencia y hace especial énfasis en errores que pueden resultar de aplicar conceptos de la lengua materna del estudiante. Un ejemplo de esto son los errores en el orden de las palabras (*word order errors*). En estos casos se indica a cuál idioma corresponde el orden utilizado y se indica también el orden correcto en inglés. Otro ejemplo de este tipo de sistemas, uno más cercano en el tiempo, es el descrito en [7] publicado en el año 2016. Este sistema, implementado para enseñar mandarín a estudiantes a través del idioma inglés utiliza *mal rules* como una de las técnicas para detectar errores. Las *mal rules* son reglas que buscan específicamente oraciones con errores gramaticales en su estructura. Estas reglas identifican dichos errores y de esta forma se despliega mensajes de ayuda al estudiante. De haber varios significados posibles, el sistema pregunta al estudiante de forma interactiva qué quiso decir, traduciendo al lenguaje de origen (utilizando traducción automática) para luego desplegar pistas sobre sus errores.

En general, GEC se formula como una tarea de corrección de oraciones. Un sistema GEC toma una oración potencialmente errónea como entrada y se espera que la transforme a su versión corregida. En la tabla 2.2 se muestra un ejemplo.

Entrada errónea	**Salida corregida**
She see Tom is catched by policeman in park at last night.	She saw Tom caught by a policeman in the park last night.

Tabla 2.2: Ejemplo GEC

Los primeros sistemas de GEC se basan en el uso de reglas que incorporan el uso de analizadores sintácticos, características lingüísticas, diccionarios y lexicones. En [21] se realiza una compilación de las diferentes técnicas utilizadas hasta el año 1992 donde también se pueden ver las primeras introducciones de técnicas de aprendizaje automático (redes neuronales) al campo. Si bien [21] se centra en la resolución de errores ortográficos, se menciona que muchas de las técnicas utilizadas para resolver estos errores son también utilizadas de forma efectiva a la hora de resolver otros tipos de errores gramaticales.

Como sucede en otras tareas de PLN, implementar un enfoque basado en reglas implica diseñar un conjunto que logre abarcar todos los posibles casos de interés y verificar a su vez que estas reglas no interfieran entre sí. Esto puede resultar en una tarea no trivial y costosa en términos de tiempo y esfuerzo. Al igual que en otras áreas de PLN, se ha adoptado el uso de técnicas estadísticas y basadas en aprendizaje automático para atacar estos problemas dados los avances en dichas técnicas y la disponibilidad de conjuntos de textos anotados.

En [1] y [36] se exploran los métodos utilizados en el área de GEC hasta el año de

publicación (2019 y 2020 respectivamente) siendo estos divididos en los siguientes grupos: basados en reglas, basados en clasificadores y basados en traducción automática. Dentro de este último grupo se diferencian a su vez los sub-grupos *Statistical Machine Translation* (SMT) y *Neural Machine Translation* (NMT).

Los métodos basados en clasificadores utilizan los *POS tags*, n-gramas y relaciones gramaticales como *features* para entrenar modelos de aprendizaje automático capaces de detectar o corregir errores. Los modelos entrenados utilizando estos métodos tienden a centrarse en un tipo de error específico, ya que identificar diferentes tipos de errores requiere el uso de diferentes *features*. Al utilizar un modelo para un único tipo de error se asume que los errores en un texto son independientes, lo cual no es siempre válido. Una posible solución en esta situación es aplicar los distintos modelos secuencialmente en forma de "cascada", corrigiendo los distintos tipos de errores a medida que son encontrados. Si bien este enfoque funciona en ciertos casos, no siempre es posible encontrar un único orden que logre corregir todos los errores.

En cuanto a los métodos basados en traducción automática, estos se utilizan para traducir secuencias de texto desde un idioma original a un idioma objetivo (por ejemplo, desde el español al inglés). Al utilizarlos dentro del área de GEC, se trata al texto con errores como el idioma original (el que se busca traducir) y el idioma objetivo es el texto correcto sin errores. En este caso y al igual que en otras áreas de PLN encontramos métodos probabilísticos para resolver este problema mediante *SMT - Statistical Machine Translation* y métodos basados en aprendizaje automático a través de *NMT - Neural Machine Translation*. Ambas técnicas necesitan conjuntos de datos anotados y mientras *NMT* logra ser más efectivo a la hora de detectar errores dependientes del contexto, *SMT* requiere menos recursos computacionales y tiempo de entrenamiento. En [15] en el año 2018 se combinó el uso de ambas técnicas (*SMT* y *NMT*) de forma efectiva logrando avanzar el estado del arte.

Tanto las técnicas de aprendizaje automático como las técnicas estadísticas requieren el uso de datos anotados. Los datos anotados utilizados en GEC generalmente se pueden dividir en dos grupos: textos con errores donde estos son identificados y marcados utilizando alguna sintaxis especial, con su corrección correspondiente (realizada por anotadores a mano) o pares de textos donde uno de los textos es el texto con errores y el otro corresponde al mismo texto luego de haber sido corregido. En este último caso los textos no poseen ninguna etiqueta o identificación de los errores.

Sobre textos anotados a mano, es interesante destacar el *Cambridge Learner Corpus*. En [27] se describe este corpus de 16 millones de palabras y creado a partir de exámenes EFL (*English as a Foreign Language*) que fueron anotados a mano. El propósito de este corpus es analizar y extraer información fácilmente sobre errores gramaticales frecuentes en estudiantes de inglés de 86 lenguas nativas diferentes. El corpus permite buscar determinada palabra y acceder a ejemplos donde dicha palabra formó parte de algún tipo de error, incluyendo la lengua nativa del estudiante. También se muestran estadísticas sobre las categorías de error donde estuvo involucrada la palabra en cuestión. Por otra parte, se puede buscar por tipo de error, mostrándose ejemplos donde fue detectado y permitiendo filtrar según la palabra involucrada, la lengua nativa y el nivel de examen donde fue encontrado determinada categoría de error.

Entre los conjuntos de pares de textos, uno de los más populares es Lang-8 Learner

Corpora[6], el cual deriva del servicio web homónimo[7]. En este sitio web, los usuarios pueden escribir textos en el idioma que están intentando aprender, mientras que usuarios que "dominan" ese idioma proveen correcciones y sugerencias sobre el texto escrito. El conjunto de datos cuenta con más de un millón de oraciones en inglés con su corrección correspondiente. Si bien este conjunto de pares de textos es el más extenso, hay que tener en cuenta que existe una gran diferencia en la calidad de las correcciones dado que no se conoce la habilidad en el lenguaje de los diferentes usuarios realizando las correcciones. Hay que tener en cuenta también que existen muchas veces también comentarios agregados dentro de las correcciones que el usuario "corrector" proporciona al usuario "estudiante". Estos comentarios son útiles en el marco de la interacción a través del sitio web, pero a la hora de utilizar los datos como pares de texto erróneo-texto correcto, incorporan ruido.

A diferencia de la traducción automática, donde abundan los datos anotados, es muy difícil conseguir grandes cantidades de textos no gramaticales debidamente anotados y sus contrapartes corregidas, lo que agrega dificultades al entrenamiento de modelos de GEC. Esto llevó a investigar el uso de técnicas para producir pares de texto de forma artificial como por ejemplo en [31], aumentando así la cantidad de datos disponibles. En este caso, se crearon 200 millones nuevos pares de oraciones. Aunque se proponen métodos para aumentar los datos y así aliviar el problema, si los datos generados artificialmente no pueden capturar con precisión la distribución de errores en datos erróneos reales, el rendimiento final de los sistemas de GEC se verá afectado. [36]

Desde el año 1999, la CoNLL (*Conference on Computational Natural Laguage Learning*) incluye una tarea compartida *shared task* que busca promover la investigación de diferentes áreas del PLN ofreciendo una forma sistemática de evaluación. La organización provee los datos de entrenamiento y de validación para cada tarea y al finalizar se presenta una publicación describiendo los resultados y técnicas exploradas por los distintos grupos participantes. Las tareas compartidas de los años 2013 y 2014 fueron sobre GEC, y en [26] y [25] podemos encontrar las publicaciones correspondientes a dichas tareas respectivamente. La tarea propuesta en el año 2013 busca corregir cinco tipos de error encontrados en los textos, mientras que en la tarea del año 2014 se busca corregir todos los distintos tipos de error encontrados en los textos (28 en total), agregando también un segundo anotador humano al conjunto de datos. En ambas ediciones los sistemas propuestos utilizan métodos híbridos compuestos de reglas para identificar errores, clasificadores individuales dedicados a un tipo de error específico, técnicas basadas en modelos de lenguaje y técnicas de traducción automática.

Actualmente existen varias herramientas comerciales para la corrección de errores gramaticales en inglés. Entre ellas *WebSpellChecker*[8] y *Grammarly*[9]. El equipo de investigadores de *Grammarly* publicó en el año 2020 el paper GECToR [28] (*Grammatical Error Correction: Tag, not rewrite*) el cual logró avanzar el estado del arte al ser evaluado sobre el conjunto de datos *CoNLL-2014 shared task* [25] y *BEA-2019 shared task* [4].

GECToR es un etiquetador de secuencias que utiliza la arquitectura de *transformers* y representa un cambio en cuanto a que una gran parte de las soluciones a los problemas de GEC eran tratados como problemas de traducción automática. Al tratar el problema como un problema de etiquetado, se evita tener que generar texto en el "idioma objetivo" lo

[6] `https://sites.google.com/site/naistlang8corpora/home/readme-en` - Último acceso: Agosto 2021
[7] `https://lang-8.com/` - Último acceso: Agosto 2021
[8] `https://www.webspellchecker.com/` - Último acceso: Febrero 2021
[9] `https://www.grammarly.com/` - Último acceso: Febrero 2021

cual simplifica el proceso de entrenamiento. Y al obtener etiquetas, es posible determinar el tipo de error encontrado, algo que no es trivial en sistemas basados en técnicas de MT.

2.3. Detección de errores de ortografía

La detección y corrección de errores de ortografía es parte integral de los editores de texto modernos y motores de búsqueda. Es útil también para corregir errores en sistemas de reconocimiento óptico de caracteres (OCR: *Optical Character Recognition*) y reconocimiento automático de texto escrito a mano.

Podemos distinguir dos tipos distintos de errores de ortografía. Uno de ellos donde la palabra que posee errores resulta en una "no-palabra", como en el caso de escribir "apple" como "aplle". El otro tipo ocurre cuando la palabra que posee errores de ortografía, resulta ser otra palabra válida del lenguaje. Por ejemplo, al intentar escribir "piece", pero escribir "peace". Estos tipos de errores necesitan del contexto en el cual la palabra fue utilizada para poder ser identificados. Si encontramos la oración: "They want a **peace** of cake.", resulta más sencillo identificar el error donde la palabra "peace" debería ser "piece".

Al primer tipo de error se lo identifica como **non-word errors**, mientras que al último se lo identifica como **real-word errors** [20]. Para detectar y corregir los errores del tipo **non-word errors** generalmente se detectan las palabras que no pertenecen a un diccionario. Es importante decidir el tamaño de diccionario a utilizar ya que un diccionario muy extenso, podrá incluir palabras cuya frecuencia en el lenguaje (o en el dominio en el cual utilizaremos el diccionario) es extremadamente baja. El problema con estas palabras, es que pueden ser iguales a la forma con error de otra palabra. Por ejemplo, consideremos la palabra "wont", cuya traducción al español es: "la forma habitual de reaccionar ante una determinada situación". Generalmente al encontrar esta palabra es más probable que haya un error de ortografía donde se intentó escribir "won't".

A la hora de corregir los errores de ortografía, generalmente se utilizan métodos de distancia de edición para encontrar la palabra más "cercana" en el diccionario. Estos métodos cuentan la cantidad de ediciones (inserciones, sustituciones y eliminaciones) requeridas para transformar una palabra. Uno de estos métodos es la distancia de Damerau-Levenshtein la cual se define recursivamente de la siguiente manera:

Sea $d_{(s,t)}(i,j)$ la distancia de Damerau-Levenshtein entre un prefijo de largo i de la palabra s y un prefijo de largo j de la palabra t.

$$d_{(s,t)}(i,j) = \begin{cases} 0 & \text{si } i = j = 0 \\ d_{(s,t)}(i-1,j) + 1 & \text{si } i > 0 \\ d_{(s,t)}(i,j-1) + 1 & \text{si } j > 0 \\ d_{(s,t)}(i-1,j-1) + 1_{(s_i \neq t_j)} & \text{si } i,j > 0 \\ d_{(s,t)}(i-2,j-2) + 1 & \text{si } i,j > 1 \text{ y } s[i] = t[j-1] \text{ y } s[i-1] = t[j] \end{cases} \tag{2.1}$$

Donde $1_{(s_i \neq t_j)}$ es una función que devuelve 0 si $s[i] = t[j]$ o 1 en cualquier otro caso.

Cada llamado recursivo de esta función se corresponde con una de las operaciones

mencionadas.

- $d_{(s,t)}(i-1,j)+1$ corresponde con eliminar un carácter de la palabra s.
- $d_{(s,t)}(i,j-1)+1$ corresponde con insertar un carácter en la palabra s.
- $d_{(s,t)}(i-1,j-1)+1_{(s_i \neq s_j)}$ corresponde con sustituir un carácter en la palabra s.
- $d_{(s,t)}(i-2,j-2)+1$ corresponde a la transposición de dos caracteres adyacentes.

A modo de ejemplo, supongamos que nos encontramos con la palabra "sitten", y queremos calcular la distancia con dos palabras ("sitting" y "kitten") pertenecientes al diccionario para realizar una sugerencia.

La distancia entre "sitten" y "sitting" es de 2, ya que:

"sitten" $\longrightarrow$ "sittin" (sustitución de **e** por **i**).

"sittin" $\longrightarrow$ "sitting" (inserción de **g** al final de la palabra).

Mientras que la distancia entre "sitten" y "kitten" es de 1, ya que:

"sitten" $\longrightarrow$ "kitten" (sustitución de **s** por **k**).

La palabra "kitten" al tener una menor distancia de edición será la palabra sugerida en este ejemplo.

Por otra parte, para corregir errores del tipo **real-word errors** se puede utilizar una versión extendida del modelo de canal ruidoso (*noisy channel model*).

La idea detrás del modelo de canal ruidoso es tratar la palabra con error de ortografía como si ésta fuera el resultado de transmitir la palabra correcta a través de un canal de comunicación ruidoso que altera la palabra (en la forma de sustituciones, y otras transformaciones) [20]. El objetivo es, entonces, construir un modelo de este canal ruidoso. Al tener este modelo, podemos pasar por el canal ruidoso todas las palabras del vocabulario, obtener los resultados, y compararlos con la palabra con error para obtener la palabra original.

La extensión del modelo de canal ruidoso a la hora de tratar con los errores del tipo **real-word errors** implica utilizar una secuencia de palabras, donde el canal ruidoso generará un conjunto de sugerencias para cada palabra de la secuencia. Este conjunto de sugerencias estará compuesto por la palabra original, más las palabras encontradas por el canal ruidoso (transformaciones de la palabra original) que pertenezcan al vocabulario. Luego se elige una sugerencia para cada palabra de la secuencia de modo que se maximice la probabilidad de la secuencia entera al utilizar n-gramas.

2.4. Modelos de lenguaje

El modelado del lenguaje es la tarea de predecir qué palabra viene a continuación o, de manera más general, es un sistema que asigna una probabilidad a una secuencia de palabras en determinado lenguaje.

Informalmente, los modelos de lenguaje buscan utilizar la noción ilustrada en la frase del profesor de Linguística inglés John Rupert Firth:

You shall know the nature of a word by the company it keeps.

(Conocerás la naturaleza de una palabra por la compañía que mantiene.)

En este sentido, un modelo de lenguaje busca predecir por ejemplo, qué palabra es más probable que aparezca a continuación de la siguiente secuencia de palabras

Los informativos muestran noticias ...

En este caso la siguiente palabra en la oración podría ser "sobre" o "intimidantes" o "enriquecedoras", pero probablemente no sea "el" o "automóvil".

La tarea de predecir la siguiente palabra o asignar probabilidades a secuencias de palabras, es esencial cuando se trabaja en el **reconocimiento del habla** (*speech recognition*) donde se debe identificar palabras en discursos ruidosos o ambiguos. También es aplicable al **reconocimiento de la escritura** (*handwriting recognition*) o en herramientas de **corrección de errores** ya sean ortográficos (*spelling correction*) o gramaticales (*grammatical error correction*).

Los modelos de lenguaje también pueden ayudar en la **producción del lenguaje**. Un ejemplo de esto son los asistentes de tipeo basados en autocompletar, que comúnmente se muestran en las aplicaciones de mensajería de texto. Son utilizados también en el área de **comunicación aumentativa y alternativa** (*augmentative and alternative communication*) para el apoyo a personas con discapacidad. [19]

Asignar probabilidades a las secuencias también es esencial en trabajos de **traducción automática**. Un modelo de lenguaje puede sugerir que una traducción es más frecuente que otra en el lenguaje destino.

Existen dos grandes grupos de modelos de lenguaje: los modelos de lenguaje estadísticos y los modelos de lenguaje neuronales. Éstos últimos siendo los representantes del estado del arte en el área. En el grupo de modelos de lenguaje estadísticos encontramos los **n-gramas**, mientras que en el grupo de los modelos neuronales encontramos implementaciones que utilizan **redes neuronales recurrentes** (*RNN: Recurrent neural networks*), **redes de memoria a corto plazo** (*LSTM: Long Short-Term Memory*) y ***Transformers***.

2.4.1. N-gramas

El modelo de lenguaje más simple es llamado *n-gramas*. Un *N-grama*, es una secuencia de n palabras: un *bigrama* es una secuencia de dos palabras como "Por favor" o "un gato" y un *trigrama* es una secuencia de tres palabras como por ejemplo "un gato negro".

Este tipo de modelos de lenguaje estiman la probabilidad de la última palabra de un n-grama dadas las n - 1 palabras previas, y también asignan probabilidades a secuencias enteras de palabras.

Para representar la probabilidad de una variable aleatoria que toma el valor de la palabra "hola", $P(X = \text{“}hola\text{”})$ utilizaremos la notación $P(\text{“}hola\text{”})$. Utilizaremos $w_1...w_n$ o $w_{1:n}$ para representar una secuencia de N palabras y $P(w_1, w_2, ..., w_n)$ para la probabilidad conjunta de $P(X = w_1, Y = w_2, ..., Z = w_n)$.

Para calcular la probabilidad de una secuencia de N palabras podemos entonces utilizar la regla de la cadena de las probabilidades:

$$P(X_1, ..., X_n) = P(X_1)P(X_2|X_1)P(X_3|X_{1:2})...P(X_n|X_{1:n-1}) = \prod_{k=1}^{n} P(X_k|X_{1:k-1})$$

y aplicando esta regla a la secuencia de palabras obtenemos:

$$P(w_1, ..., w_n) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(w_n|w_{1:n-1}) = \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

Esto nos permite calcular la probabilidad de manera condicional, pero aún no es sencillo determinar $P(w_n|w_{1:n-1})$ para secuencias largas de palabras. Aquí es donde podemos utilizar la propiedad de Márkov (*Márkov assumption*), en este caso en particular afirmando que la probabilidad de una palabra queda determinada por las palabras inmediatamente anteriores a ésta. Formalmente,

$$P(w_n|w_{1:n}) \approx P(w_n|w_{n-N+1:n-1})$$

donde N ($N > 0$) es la cantidad de palabras "del pasado" que utilizamos para calcular la probabilidad. Si $N = 2$, es un modelo de bigramas y utilizamos sólo la palabra inmediatamente anterior. Mientras que si $N = 3$, hablamos de un modelo de trigramas que utiliza las dos palabras anteriores.

Para estimar las probabilidades de los distintos n-gramas se utiliza MLE (*Maximum Likelihood Estimation* - o Estimación por máxima verosimilitud) sobre un corpus de texto. Utilizando bigramas a modo de ejemplo, para calcular la probabilidad de un bigrama determinado por las palabras a y b, contamos la cantidad $C(ab)$ de ocurrencias en el corpus y normalizamos utilizando la cantidad $C(a)$ de unigramas correspondientes a la palabra a obteniendo asi valores entre 0 y 1.

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

De esta manera los modelos de lenguaje basados en *n-gramas* pueden determinar la probabilidad de una palabra utilizando las $n - 1$ palabras anteriores, o la probabilidad de una secuencia de n palabras.

2.4.2. Modelos neuronales

Estos modelos utilizan redes neuronales como un clasificador probabilístico para calcular la probabilidad de la posible siguiente palabra de una secuencia dadas las n palabras anteriores. [19]

Una red neuronal es una técnica de aprendizaje automático inspirada en el funcionamiento de las neuronas en el cerebro humano. Se busca crear redes compuestas de unidades de cómputo simple (neuronas), organizadas en capas, ajustando el cómputo de cada neurona durante el entrenamiento para que la red en su conjunto logre "aprender" a ejecutar una tarea determinada. En la figura 2.6 se muestra una red neuronal de tipo *feedforward* donde el cómputo se calcula de forma iterativa de una capa a la siguiente. El uso moderno de redes neuronales es frecuentemente llamado *deep learning* por la gran cantidad de capas que forman dichas redes.

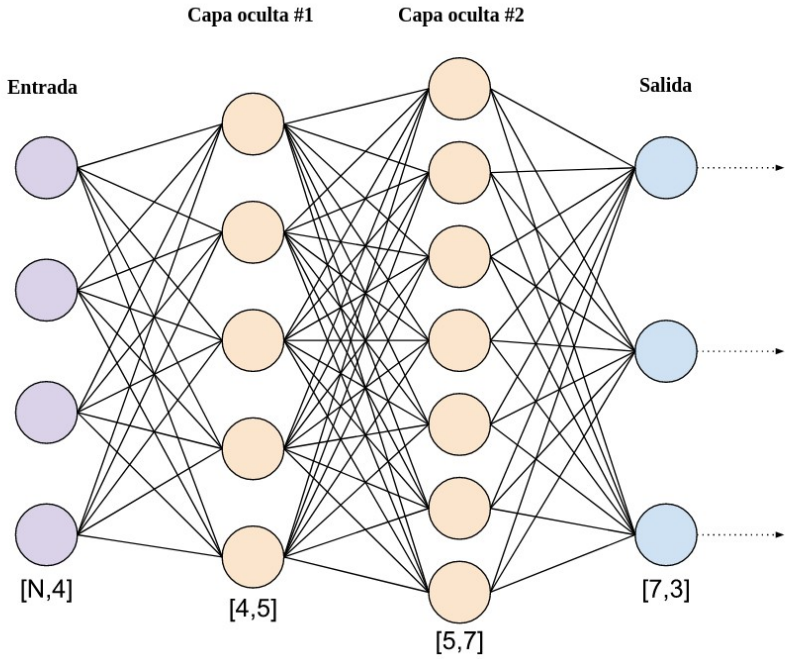


Figura 2.6: Representación de la conexión en red de las neuronas en una red neuronal de tipo *feedforward*.

Los modelos de lenguaje neuronales presentan ciertas ventajas sobre sus contrapartes basados en *n-gramas*. Pueden utilizar historias más largas y pueden generalizar sobre contextos de palabras similares. Para un conjunto de entrenamiento de tamaño dado, un modelo de lenguaje neuronal tiene un acierto de predicción mucho mayor al de un modelo de lenguaje basado en *n-gramas* [19]. Por otra parte los modelos de lenguaje neuronales requieren un mayor tiempo de entrenamiento, por lo cual, dependiendo del problema a resolver, un modelo de lenguaje basado en *n-gramas* puede ser una mejor alternativa.

BERT

Presentado en [9] por el equipo de Google AI Language[10] en el año 2018, BERT, *Bidirectional Encoder Representations from Transformers* es un modelo de representación del lenguaje, pre-entrenado utilizando el BookCorpus [38] (800 millones de palabras) y Wikipedia en inglés (2500 millones de palabras, ignorando tablas, listas y cabezales) y logró al momento de ser presentado, avanzar el estado del arte en 11 tareas relacionadas

[10] `https://research.google/teams/language/` - Último acceso: Febrero 2021

al PLN entre ellas *Question Answering* y *Language Inference*.

BERT implementa en el área del PLN un paradigma utilizado en el área de *computer vision*. En este paradigma, se utilizan modelos pre-entrenados sobre un gran conjunto de (pre-)entrenamiento de imágenes, los cuales logran captar detalles como bordes, formas y objetos para luego utilizar esta información junto con un conjunto de entrenamiento (de menor tamaño que el de pre-entrenamiento y con un dominio determinado) para resolver una tarea específica.

Es de esta misma manera que podemos utilizar el modelo pre-entrenado de BERT para resolver distintas tareas de PLN, sin tener que invertir el tiempo y los recursos en entrenar modelos específicos. Así se pueden utilizar las representaciones aprendidas, junto con los datos de entrenamiento correspondientes a la tarea a resolver, para entrenar un modelo que resuelva dicha tarea.

Sabemos que un modelo de lenguaje busca indicarnos la probabilidad de la posible siguiente palabra en una secuencia de palabras. BERT nos da representaciones (en forma de vectores) de las palabras en la secuencia de acuerdo a su contexto, por lo cual decimos que BERT es un modelo de representación del lenguaje en lugar de un modelo de lenguaje. De todas formas, es sencillo utilizar a BERT como un modelo de lenguaje ya que una de las dos tareas sobre las que fue pre-entrenado, es el modelado de lenguajes.

Las dos tareas (no supervisadas) sobre las cuales se realizó el pre-entrenamiento son: modelo de lenguaje enmascarado (*Masked Language Model*) y predicción de la siguiente oración (*Next Sentence Prediction*).

La tarea de modelo de lenguaje enmascarado se inspira en la tarea de Cloze [34] donde se "enmascara" o remueve una palabra de una secuencia y se busca identificar la misma utilizando el contexto dado en el resto de la secuencia. Esto asegura la bi-direccionalidad del modelo de lenguaje ya que es capaz de utilizar el contexto anterior y posterior a la palabra enmascarada para tomar una decisión.

La tarea de predicción de siguiente oración busca preparar al modelo para enfrentar una situación común en varias tareas de PLN, determinar si existe una relación entre dos oraciones. Esto es útil en tareas como *Question Answering* y *Natural Language Inference*. Para esta tarea se entrena utilizando pares de oraciones, donde 50 % de los pares son oraciones contiguas en el corpus de entrenamiento y 50 % son dos oraciones elegidas al azar.

Transformers

BERT utiliza una arquitectura llamada *Transformers*, introducida en [35]. Esta arquitectura presenta ciertas ventajas con respecto a las arquitecturas que utilizan Redes Neuronales Recurrentes (RNNs) entre las cuales podemos mencionar que son fácilmente paralelizables, logrando reducir de manera considerable los tiempos de entrenamiento.

Las arquitecturas que utilizan RNNs procesan una palabra a la vez, lo que genera una noción de "orden" entre las palabras de la secuencia en el modelo. Al procesar la secuencia completa como entrada, como en el caso de los *transformers*, es necesario incorporar esta noción de "orden" de alguna manera. Esta arquitectura propone incorporar la información posicional de cada palabra en la representación de entrada de cada secuencia. El hecho de

poder procesar la secuencia completa es lo que permite que BERT sea bi-direccional. Los mecanismos de atención y auto-atención de los *transformers* logran identificar de mejor manera relaciones entre elementos distantes de las secuencias, algo que resulta difícil al utilizar arquitecturas basadas en RNNs (por ejemplo, ver el *vanishing gradient problem* [17]).

Los *transformers*, a diferencia de las arquitecturas basadas en RNNs, tienen un tamaño de entrada fijo. En el caso de BERT por ejemplo, la secuencia de entrada tiene que ser menor o igual a 512 *tokens*. Al intentar procesar secuencias más extensas que el límite, deberán ser segmentadas y procesadas de forma separada, lo que provoca una fragmentación de contexto. Cualquier contexto "aprendido" del primer segmento, no será utilizado a la hora de procesar el segundo segmento (y viceversa).

Problemas con modelos de lenguaje extensos

Es importante tener en cuenta que los modelos de lenguaje, especialmente aquellos entrenados con un gran volumen de texto pueden presentar ciertos problemas.

Modelos de lenguaje utilizados para generar texto pueden incurrir en el uso de lenguaje tóxico y abusivo, incluso en casos donde el texto utilizado como entrada (*input*) no contenía lenguaje tóxico [19][12]. El uso extensivo de texto de ciertos sitios web, como *Reddit*[11], donde los autores de los textos tienden a ser varones jóvenes genera un sesgo demográfico que puede llevar al modelo a tener problemas para generar tópicos y perspectivas de poblaciones con poca representación.

En Carlini et al. (2020) [19][5] se muestran problemas relacionados a la privacidad, y cómo ciertos modelos de lenguaje pueden ser explotados para obtener información (nombres, teléfonos y direcciones) perteneciente a los datos de entrenamiento. Resulta importante entonces, en el caso de utilizar un modelo de lenguaje ajeno, el poder conocer qué tipo de datos fueron utilizados durante su entrenamiento.

2.5. Aprendizaje automático

El Aprendizaje Automático es una rama de la IA, cuyo objetivo es desarrollar programas que mejoren su desempeño en una tarea a través de la experiencia. Se aprende a partir de un conjunto de ejemplos o instancias que ya están representadas a través de atributos (*features*) o utilizando ejemplos a los que la hace falta preprocesamiento para ser convertidos a números (por ejemplo en el caso de imágenes o texto).

En el aprendizaje supervisado, los algoritmos trabajan con datos etiquetados, intentando encontrar una función que, dadas las variables de entrada, les asigne la etiqueta de salida adecuada. El aprendizaje supervisado se suele usar en problemas de **clasificación** donde el algoritmo ajusta un modelo a los datos obteniéndose un clasificador el cual estimará valores para nuevos elementos.

[11] `https://www.reddit.com/` - Último acceso: Marzo 2021

2.5.1. Clasificación

Existen etapas o fases comunes para resolver problemas de clasificación: [23]

- **Preprocesamiento.** El conjunto de datos del que generalmente se dispone surge de sensores, datos ingresados por humanos o fuentes diferentes, por lo tanto se debe hacer una limpieza y convertir a un formato estándar. Puede que, para ser más eficientes en los tiempos de aprendizaje, sea necesario agrupar datos, eliminar atributos o reducir el número de instancias, buscando no perder información.

- **División del conjunto de datos.** Inicialmente el conjunto de datos se separa en dos conjuntos: uno de entrenamiento y otro de evaluación. Se debe asegurar, para evitar el sobreajuste (*overfitting*), que la evaluación del modelo se realice en un conjunto de datos distinto a aquel sobre el cual se entrenó. Usualmente se divide como 80 %-20 %, o 70 %-30 %.

  Si se quiere ajustar los parámetros del modelo, no es conveniente hacerlo en el conjunto de evaluación ya que podríamos estar sobreajustando, nuevamente. Una opción es separar una parte del conjunto de entrenamiento para utilizarlo en esa etapa de validación y la otra opción es utilizar validación cruzada (*cross-validation*). En la validación cruzada, se divide el *dataset* de entrenamiento en k partes, y se utilizan *k-1* partes para entrenar, y la restante para evaluar el modelo. Este proceso se repite cambiando la parte elegida.

  Un paso más, especialmente importante cuando las clases objetivo están desbalanceadas (es decir, hay muchos más ejemplos de una clase que de otras), es estratificar: elegir las instancias en cada una de las subclases, obligando a que la proporción sea la misma en el corpus de entrenamiento y en el de evaluación.

- **Entrenamiento.** Durante el entrenamiento, se utilizan los datos del conjunto de entrenamiento, y un algoritmo de aprendizaje, para generar un clasificador. Los algoritmos de entrenamiento tienen usualmente hiperparámetros que deberían ajustarse para obtener mejores resultados.

  También se realiza una selección de atributos que "valen la pena" para la tarea de clasificación que se intenta hacer, es decir, se selecciona aquellos que no sean redundantes o irrelevantes.

- **Evaluación.** El paso final es evaluar la *performance* del modelo obtenido sobre un conjunto de datos no visto previamente. Lo más sencillo es estimar el acierto (*accuracy*): mide el porcentaje de casos que el modelo ha acertado. El problema con el acierto y el error es que no tienen en cuenta el comportamiento en las distintas clases.

  Con la métrica de precisión (*precision*) se puede medir el procentaje de positivos que detecta el modelo correctamente. Muestra qué tan bueno es el clasificador cuando predice que un ejemplo es positivo.

  La métrica de exhaustividad (*recall*) informa sobre la proporción que el modelo es capaz de identificar de los positivos existentes.

  La medida F1 (*F1-measure*) se utiliza para combinar las medidas de *precision* y *recall* en un sólo valor. Esto es práctico porque hace más fácil el poder comparar el

rendimiento combinado de las medidas *precision* y *recall* entre varias soluciones. F1 se calcula haciendo la media armónica entre ambas medidas.

Una forma muy útil para intentar entender el comportamiento del clasificador es estudiar la matriz de confusión, especialmente cuando se tiene más de dos clases. Las filas de la matriz representan las instancias pertenecientes en el conjunto a cada clase, mientras que las columnas indican cómo fueron clasificadas por el modelo. En la diagonal principal quedan los elementos correctamente clasificados.

2.5.2. Representación de textos

Cuando se trata de textos, para obtener atributos necesarios para la tarea de clasificación, muchas veces se emplea el método tradicional *Bag of Words* (BoW). A partir de un vocabulario (lista de palabras del lenguaje), se construye un vector con un atributo por palabra. Para completar este vector se puede optar por uno de los siguientes enfoques:

- Valor 1 indicando que la palabra existe en el texto, 0 que no (*one-hot encoding*).
- Cantidad de ocurrencias de la palabra en el texto (eventualmente normalizada, dividiendo sobre el total de palabras del documento)
- tf-idf *term frequency-inverse document frequency*: pondera la frecuencia de la palabra viendo qué tan común es en general (un valor alto indica que la palabra es común en el texto de la instancia, pero rara en el *dataset*)

En la figura 2.7 [20] se muestra visualmente un ejemplo donde se hace uso de la frecuencia de cada palabra en el texto.

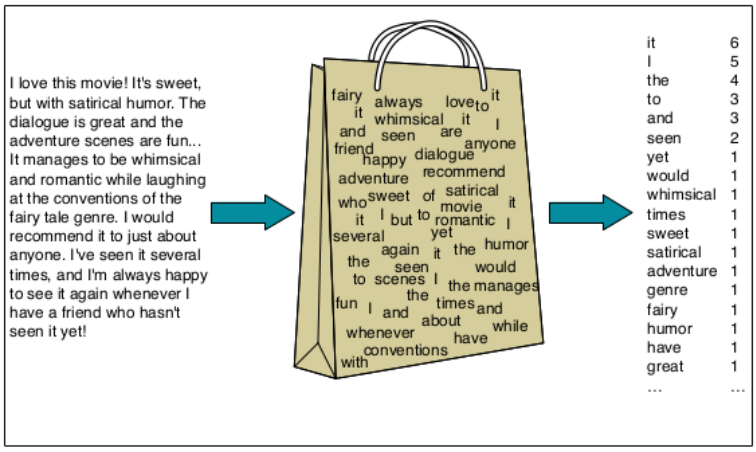


Figura 2.7: Ejemplo de BoW

Dentro de las limitaciones que presenta esta forma de representar textos, podemos destacar el hecho de carecer de información sobre las relaciones entre palabras a lo largo del texto. Además, los vectores resultantes, cuyo largo corresponderá con la cantidad de elementos en el vocabulario utilizado, tendrán gran cantidad de entradas en 0.

Con el enfoque de *Word Embeddings* se introduce la codificación tanto de la semántica como de la relación entre palabras. La idea de ésta técnica es representar cada palabra

con un vector, un punto en un espacio semántico multidimensional. [19] Los vectores que representan palabras con significados similares se ubican más cerca entre sí, y el significado de cada palabra viene dado por su respectivo entorno.

Contextual Embeddings como BERT (2.4.2), van más allá al ofrecer un *embedding* que representa el significado de la palabra en su contexto textual. Por ejemplo, la palabra "mouse" cambia su significado según el contexto donde se encuentre; puede estar refiriendo al animal o al dispositivo de la computadora.

Capítulo 3

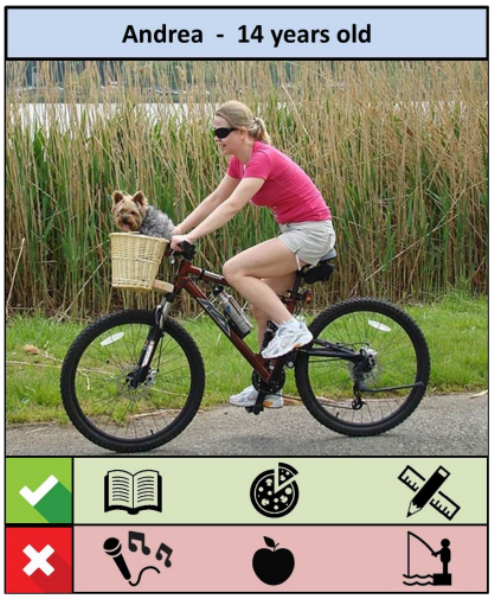
Descripción y análisis de los datos

En el presente trabajo, los datos utilizados para el desarrollo del corrector automático y posterior clasificador son textos escritos por niños de Educación Primaria. Dichos textos son parte de una prueba adaptativa de inglés propuesta en 2017 en el marco del programa educativo "Ceibal en inglés"[1] para la enseñanza del idioma mencionado a niños de 4to, 5to y 6to año de Primaria y a sus maestros.

La prueba constaba de varios ejercicios evaluando comprensión lectora (*Vocabulary*, *Reading* y *Grammar*), continuaba por comprensión auditiva (*Listening*) y finalizaba con la consigna de producción escrita (*Writing*) dividida en dos partes. La primera parte consistía en seis ejercicios relacionados a una imagen, donde debían completar oraciones con una, dos o tres palabras. En el presente trabajo se utiliza la segunda parte del *Writing* donde se le propone a los niños describir otra imagen mencionando características de la persona que allí aparece.

En la figura 3.1 se muestra la parte 2 de la propuesta de writing para la Prueba Adaptativa de Inglés de 2017.

[1] `https://ingles.ceibal.edu.uy/`



Mira la imagen. Esta es Andrea. Escribe en inglés en la casilla todo lo que puedas sobre ella. Presta atención a la foto y a los dibujos: puedes escribir sobre su edad; su aspecto físico; la ropa que está vistiendo en la foto; lo que le gusta y no le gusta; lo que tiene y lo que no tiene; lo que puede y no puede hacer; sus rutinas; etc.

Figura 3.1: Parte 2 del ejercicio de *Writing*

Las pruebas fueron corregidas por profesores de inglés y se les asignó un puntaje entre 0 y 6 según los criterios de corrección descriptos en el apéndice A.

El conjunto de datos está compuesto por 65.528 textos con la distribución de puntajes que se muestra en la tabla 3.1. Analizando dicha tabla se puede observar que el corpus está desbalanceado: más del 50 % de los ejemplos pertenecen a las puntuaciones 0 y 1, y la representación de los puntajes más altos es muy poca.

Puntaje	**Cant. de textos**	**Cant. de textos vacíos**
0	23.002	9256
1	13.034	1606
2	17.669	615
3	10.295	14
4	1350	-
5	135	-
6	13	-
Sin nota	30	6
Total	65.528	11.497

Tabla 3.1: Distribución de puntajes en el corpus

Otro dato a tener en cuenta es que hay 11.497 ejemplos totalmente vacíos pertenecientes a los puntajes 0, 1, 2 y 3. Esto ocurre ya que el puntaje asignado a la prueba depende no sólo de la parte de desarrollo, sino del desempeño en los otros 6 ejercicios pertenecientes

a la parte 1 de la prueba de *Writing*. En la tabla 3.1 también se muestra la distribución de textos vacíos según el puntaje.

En la tabla 3.2 se muestran ejemplos de los textos para cada puntaje. Se puede observar que los textos de puntajes más altos tienden a ser más extensos y logran generar construcciones gramaticales más complejas, así como que la cantidad de palabras en español presentes en los textos de nota más baja tiende a ser más alta.

Puntaje	Ejemplo
0	le gusta leer comer pipza y escribir lo que no le gusta es cantar comer fruta y pescar
1	i like reading,pizza and rite. i don’t like apple,to sing and fish his she andrea 14 years old
2	she wears a pink shirt and jeans shorts he likes to ride a bicycle
3	She has got a dog. She has got a glass in her face. She has got a bike. She drive in a bike. She like read and draw. She like eat pizza. She hate sing. She doesn't like eat apples.
4	Andrea is 14 years old, she is a blondy and athletic girl. She is wearing a pink t-shirt, a white short and sunglasses. She is reading a bike whit her pet, a little dog. She likes eat pizza but doesn't like apples. She has a lot of books because she likes to read. Andrea studies from monday to friday. She doesn't like to fish because it's boring, she doesn't know how to sing
5	She is Andrea. She is 14 years old. She tall and thin. She has blonde, long hair. She is wearing white trainers, beige shorts, a pink blouse and sunglasses. She is riding a bike. She likes reading books, eating pizza and geometry. She doesn't like singing, eating apples and fishing. She has a pet. It's a dog. She loves it. She hasn't got a car. She can ride a bike but she can't fly. She gets up early, has breakfast and ride a bike. After that she has a bath and watch tv. Then she has lunch and goes to high school. After high school she goes to hockey classes. After she has a bath again, does her homework and goes to bed. She lives in a big house with his mother, father and sister. She loves her family and she is very happy.
6	She is Andrea, she is fourteen years old. She's wearing a pink t-shirt, and a short of jean She is riding her bike with her dog, she likes reading books, she likes eating pizza, and she likes maths. She doesn't like singing, eating apples and fishing She's got a dog but she doesn't have a cat. She doesn't look like a professional bike riding, and she isn't fat but she isn't thin. Her bike is brown and black and her dog is gray and brown, her dog is super cute, I want to be the owner of that dog, but her dog isn't like mine (Aurora) mine is cuter than hers. She's got yellow hair and a black glasses, she is riding her bike in a quiet place, like in a countryside, behind her is a big lake.

Tabla 3.2: Ejemplos por puntaje

3.1. Muestreo de datos de desarrollo y evaluación

Con el objetivo de crear un corrector automático de errores basado en heurísticas, se crearon dos subconjuntos de datos a partir del conjunto original. A estos subconjuntos, los llamaremos "desarrollo" y "evaluación". El muestreo de desarrollo fue utilizado para "censar" los tipos de errores cometidos por los estudiantes, decidir los tipos de errores sobre los cuales se construyó el corrector y encontrar las mejores estrategias para corregir dichos errores. Una vez finalizado este proceso se evaluó el corrector sobre ambos muestreos, el de desarrollo y el de validación, para comparar los resultados.

3.1.1. Muestreo de desarrollo

El subconjunto "desarrollo" consta de 53 textos elegidos al azar, donde se obtuvieron diez textos de cada puntaje del 1 al 5, y tres textos con puntaje 6. Se utilizó una cantidad reducida de textos con puntaje 6 debido a la poca cantidad que existen en el conjunto original.

Este subconjunto fue corregido de manera manual para detectar los tipos de errores y las frecuencias de los errores cometidos por los estudiantes en la producción escrita en inglés. En la columna "Desa." de la tabla 3.3 se muestran los resultados de esta corrección.

Para el diseño de las heurísticas de detección y corrección de errores, se tomaron como más prioritarios los errores más frecuentes en este muestreo.

3.1.2. Muestreo de evaluación

Con el fin de validar el corrector automático luego de desarrollado y así poder evaluar su funcionamiento y calidad, se seleccionó un muestreo de "evaluación". El subconjunto "evaluación" consta de 42 textos elegidos al azar, donde se obtuvieron diez textos de cada puntaje del 2 al 5, y dos textos con puntaje 6.

Este subconjunto fue también corregido de manera manual. La corrección en esta instancia fue efectuada por los tutores del proyecto, los cuales fueron instruídos en los tipos de errores que debían ser corregidos. La cantidad de errores encontrados por los tutores se muestran en la columna "Eval." de la tabla 3.3.

Como puede observarse en la tabla 3.3 el ranking de tipos de error en desarrollo y en evaluación es similar para gran parte de las categorías de error. Teniendo en cuenta esta observación, se realizó una selección de los tipos de error más frecuentes para trabajar en ellos.

Error	Descripción	Ejemplo	Desa.	Eval.
Spelling	Falta de ortografía	✗ reding ✓ reading	84	69
Subject-verb agreement	Sujeto y verbo no coinciden en número	✗ She have a dog ✓ She has a dog	42	15
Beginning of sentence capitalization	Comienzo de sentencia sin mayúscula	✗ she is andrea ✓ She is Andrea	39	68
Pronoun reference	Pronombre incorrecto	✗ She likes riding in your bike with your little dog ✓ She likes riding in her bike with her little dog	26	4
Verb form	Forma del verbo incorrecta	✗ She likes sing ✓ She likes singing	24	41
Missing subject	Falta sujeto en la sentencia	✗ She has blond hair, is wearing a pink sweater and beige shorts ✓ She has blond hair, she is wearing a pink sweater and beige shorts	15	19
Proper noun capitalization	Nombres propios sin mayúscula	✗ She is andrea ✓ She is Andrea	14	5
Noun number	Número incorrecto de sustantivo	✗ She likes apple ✓ She likes apples	11	6
Wrong article or determiner	Artículo o determinante innecesario o faltante	✗ and a white trousers ✓ and white trousers	7	14
Pronoun "I" capitalization	Pronombre "I" sin mayúscula	✗ i think she is an un-healthy girl ✓ I think she is an un-healthy girl	6	0
Adjective order	Ubicación incorrecta del adjetivo	✗ She has a t-shirt pink ✓ She has a pink t-shirt	4	0
Contraction	Contracción incorrecta	✗ doesnt ✓ doesn't	3	0
Missing verb	Hace falta el verbo en la sentencia	✗ She 14 years old ✓ She is 14 years old	2	3
Wrong verb	Verbo utilizado incorrecto	✗ She has 14 years old ✓ She is 14 years old	2	10
Other errors	Otros errores	✗ Finally she goes to bed at 0:00 a.m. clock ✓ Finally she goes to bed at 0:00 a.m.	24	23
		Total	303	277

Tabla 3.3: Tipos de errores y su representación en cada conjunto

Capítulo 4

Corrector

4.1. Funcionamiento general

El corrector automático recibe como entrada un texto y retorna una versión marcada y una versión corregida del texto recibido. El texto marcado es el texto original donde se utilizan etiquetas similares en estilo a etiquetas XML que indican el tipo de error y la ubicación del error encontrado. Por otro lado, el texto corregido es el resultado de aplicar las correcciones correspondientes a los errores encontrados por el corrector automático. En la figura 4.1 hay un esquema del corrector y sus diferentes componentes.

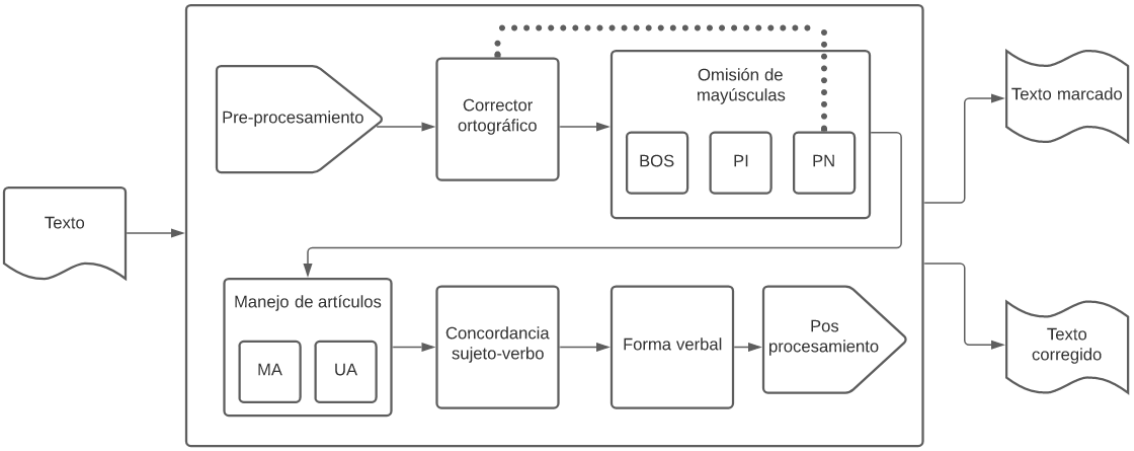


Figura 4.1: *Pipeline* del corrector.

El texto de entrada se pre-procesa antes de realizar la detección/corrección de errores. El pre-procesamiento consiste en la limpieza del texto, detección de anomalías (como se verá en la sección 4.2.1) y división del texto en oraciones y *tokens*.

Una vez finalizado el pre-procesamiento, se procede a corregir primero los errores de ortografía para poder hacer una buena detección del resto de los errores. Al encontrar los errores de ortografía se verifica si las palabras candidatas a errores son nombres propios antes de marcarlas como error; esto está representado en el esquema por la línea punteada

hacia el sub-componente **PN** (*Proper Name*).

Luego se procede a verificar los errores de omisión de mayúscula mediante los componentes **BOS** (*Beginning Of Sentence*), **PI** (*Pronoun I*), **PN** (*Proper Name*) que verifican el uso de mayúsculas al comienzo de las oraciones, en el uso del pronombre "I" y el uso de los nombres propios respectivamente.

Se procede verificando el uso de artículos, específicamente la falta de ellos en el componente **MA** (*Missing Article*) y el uso innecesario en el componente **UA** (*Unnecessary Article*).

Se continúa el procesamiento en forma de cascada pasando por los componentes que identifican los errores de concordancia sujeto-verbo y los errores de forma verbal. Al finalizar la pasada por los componentes de detección y corrección de errores, el componente de pos-procesamiento se encarga de preparar la salida concatenando los *tokens* y las oraciones del texto marcado y del texto corregido.

4.2. Pre-procesamiento

Buscando evitar errores en el procesamiento de cada texto, se trabajó previamente en el texto con el fin de estandarizarlo. Los textos claramente no tienen un formato estándar en la redacción teniendo en cuenta que son textos escritos por niños que en general, recién están teniendo un primer acercamiento a este nuevo lenguaje.

Se observó una gran variabilidad en cuanto al uso de signos de puntuación donde muchas veces las oraciones son separadas por comas o saltos de línea. También es muy variable el uso de comillas y apóstrofos, en especial para las contracciones como "don't" que se observó escrita de distintas formas: "dont", "don´t", "don t", "don't", "don‘t". Asimismo se constató que la hora aparece escrita en muchos formatos distintos.

4.2.1. Limpieza del texto

Como es de esperarse por ser textos escritos por niños de Primaria que aún no tienen buen manejo del teclado del computador, se encontraron espacios innecesarios entre palabras o falta de los mismos en algunos casos.

La limpieza del texto incluyó:

- Eliminación de espacios innecesarios dejando un único espacio entre palabras.
- Incorporación de espacios luego del signo de puntuación final y de las comas.
- Estandarización de las distintas formas de expresar "A.M." y "P.M." las cuales fueron encontradas en las formas "a.m.", "A.M.", "a. m.", "A.M", entre otras. El estándar elegido fue "AM" y "PM", de forma de evitar utilizar puntuación que pueda confundir al segmentador a la hora de determinar las oraciones y *tokens*. Una vez finalizada la segmentación, se transforman nuevamente a la forma "a.m." y "p.m." respectivamente.

4.2.2. Segmentador

Se implementó un segmentador de texto con el objetivo de separar el texto original en oraciones y *tokens*, de modo que su procesamiento sea más sencillo, utilizando el concepto de "divide y vencerás".

En la exploración inicial se pudo observar que en general los niños escriben oraciones separadas por saltos de línea y en los mejores casos con signo de puntuación al final. No se observó la presencia de oraciones escritas en dos líneas. Es por esto que el segmentador primero separa las oraciones según los saltos de línea y luego a cada una de ellas se las segmenta en oraciones más pequeñas utilizado el segmentador de NLTK[1].

4.3. Marcado del texto

El marcado se realiza mediante tags similares a los tags XML. Los tags asignados por el corrector automático encierran la palabra del texto en la cual se encontró el error. Cuando el tipo de error involucra más de una palabra, se establece una convención para determinar la palabra a encerrar con los tags; esto evita establecer tags en más de una palabra, situación que podría llevar al solapamiento de tags.

En la tabla 4.1 se muestran algunos ejemplos de marcado correspondientes a cada tipo de error. Además de la etiqueta del tipo de error y a modo de mostrar una posible corrección de la palabra en cuestión, los tags incluyen el parámetro "corr" que se agrega en todos los casos a excepción del tag de error de ortografía donde se toma la palabra correcta como la primera en la lista de sugerencias "sug". El parámetro "corr" será el utilizado para armar la salida correspondiente al texto corregido. En la tabla 4.1 no se incluyó el parámetro mencionado para simplificar el cuadro.

Error	**Tag**
Ortografía	She has got medium <E_C-Spell> legth </E_C-Spell>, blonde hair.
Mayúscula al comienzo de la oración	<E_C-BOS> she </E_C-BOS> is Andrea.
Mayúscula en pronombre "I"	She is a 14 years old, <E_C-PI> i </E_C-PI> have a dog
Mayúscula en nombres propios	This is <E_C-PN> andrea </E_C-PN>
Forma del verbo	Andrea likes to ride a bike and <E_VF> eat </E_VF>
Concordancia sujeto verbo	She <E_SVA> have </E_SVA> 14 years old
Artículo faltante	She doesn't like eating <E_MA> apple </E_MA>
Artículo innecesario	She likes reading a <E_UA> books </E_UA>

Tabla 4.1: Ejemplo de marcado para cada tipo de error

[1] `https://www.nltk.org/`

4.4. Detección de errores

Los tipos de error seleccionados para marcar en los textos son: errores de ortografía, de omisión de mayúscula, de concordancia sujeto-verbo, de forma del verbo y de artículos.

4.4.1. Ortografía

Este es el primer tipo de error que se procesa en el *pipeline* de detección de errores. Esto se debe a la necesidad de detectar y corregir los errores ortográficos para así mejorar la calidad del texto y por ende la detección de los restantes tipos de error.

Dentro de los errores ortográficos encontrados en el muestreo de desarrollo, podemos distinguir dos categorías diferentes. La primera categoría corresponde a errores donde el estudiante escribe una palabra de manera incorrecta, como por ejemplo:

*Andrea is **wering** a **pinck** t-shirt and shorts.*

En esta oración, tanto la palabra "wering" como "pinck" fueron escritas de manera incorrecta y deberían ser "wearing" y "pink" respectivamente. Esto puede suceder porque el estudiante no conoce la forma correcta de escribir dicha palabra o debido a un error de tipeo, estos errores son los identificados como **non-word errors** en la sección 2.3.

La segunda categoría representa errores donde el estudiante comete un error al escribir la palabra, pero la palabra resultante es a su vez una palabra válida en inglés, como en los ejemplos:

*... those **hare** the things she does not like to do.*

*She doesn't like apples and **finishing***

Aquí podemos ver que la palabra "hare" debería ser "are". Este tipo de error es diferente del anterior porque "hare" es una palabra correcta en inglés, la cual significa "liebre" en español. En el segundo ejemplo, la palabra "finishing" debería ser "fishing". Este tipo de errores son los identificados como **real-word errors** en la sección 2.3.

El primer tipo de error, es decir, las palabras escritas de manera incorrecta, dominan la cantidad de errores ortográficos en el muestreo de desarrollo, siendo más del 93 % del total. Por esta razón y porque es muy costoso reconocer errores del segundo tipo, el corrector ortográfico se ocupa sólo de los errores identificados como **non-word errors**.

Una vez que una palabra es detectada como un error de ortografía, es necesario realizar un último chequeo. Debemos identificar si la palabra se corresponde con un nombre propio en inglés o en español. No realizar este chequeo previo implicaría que en el caso de encontrar un nombre propio en un texto, como por ejemplo "Paco", al no ser detectado como una palabra correcta en inglés por los métodos y diccionarios utilizados por el corrector ortográfico, sea marcado como un error de ortografía, generando falsos positivos en el corrector.

Para realizar este chequeo se utiliza el sub-componente **PN** del componente **Omisión de mayúsculas** del corrector, que utiliza los recursos descritos en la sección 4.4.2.

4.4.1.1. Correctores ortográficos

Para detectar y corregir los errores ortográficos en los textos, se exploraron distintas heurísticas que involucran el uso de librerías de *spellchecking* disponibles en Python. Las librerías utilizadas fueron las que se describen a continuación.

Hunspell

Corrector ortográfico *open-source* utilizado en conocidos proyectos de software como: Firefox, OpenOffice, LibreOffice, Google Chrome y Opera, entre otros. Hunspell [2], creado originalmente para el lenguaje húngaro, fue adaptado y funciona actualmente en múltiples idiomas. Utiliza reglas para remover prefijos y sufijos, buscando la raíz de la palabra en un diccionario. Recomienda palabras candidatas utilizando *n-gramas* y datos de pronunciación.

Spellchecker

Implementación simple de un corrector ortográfico basado en el concepto desarrollado por Peter Norvig[3]. Utiliza la distancia de edición de Damerau-Levenshtein para encontrar permutaciones con la palabra original. Esta implementación genera todos los posibles términos con una distancia de edición máxima de 2 (inserción, reemplazo, eliminación y transposición de caracteres adyacentes) a partir de la palabra original para luego buscarla en el diccionario.

Si la palabra original se encuentra en el diccionario, entonces dicha palabra no tiene error. En caso de no encontrarse, los términos generados que aparecen en el diccionario son los candidatos a la corrección de la palabra original.

Para una palabra de largo n, con un alfabeto de largo a y una distancia de edición $d = 1$, habrá n eliminaciones, $n-1$ transposiciones, $a*n$ reemplazos y $a*(n-1)$ inserciones. Esto hace un total de $2n + 2an + a - 1$ términos para buscar. Este enfoque depende del alfabeto del lenguaje utilizado, el cual en nuestro caso no es extenso, pero puede ser un problema al utilizar esta técnica sobre otro idioma (el idioma chino tiene 70000 caracteres Unicode Han por ejemplo).

Symspell

Este enfoque, propuesto por Wolf Garbe[4] también se basa en la distancia de edición de Damerau-Levenshtein, pero se enfoca únicamente en las operaciones de eliminación de caracteres. El algoritmo que utiliza este corrector se llama *Symmetric Delete Spelling Correction*. Se comienza con un paso de pre-cómputo el cual debe ser realizado una única vez. Durante este paso, se calcula y se almacena la distancia de edición (solo eliminando caracteres) de todos los términos formados a partir de todas las palabras que componen

[2] `http://hunspell.github.io/` - Último acceso: Junio 2020

[3] `https://norvig.com/spell-correct.html` - Último acceso: Junio 2020

[4] `https://wolfgarbe.medium.com/1000x-faster-spelling-correction-algorithm-2012-8701fcd87a5f` - Último acceso: Junio 2020

el diccionario. Todos estos términos generados son incorporados al diccionario al finalizar el pre-cómputo.

Para verificar una palabra con el corrector, se generan los distintos términos eliminando caracteres de dicha palabra y se busca si algunos de los términos forma parte del diccionario.

Si la palabra original se encuentra en el diccionario (y por lo tanto posee una distancia de edición de cero) entonces no hubo error ortográfico, en caso contrario se retornan todos los términos que se encuentran en el diccionario y que tienen la distancia de edición máxima determinada.

Con este enfoque, dada un palabra de largo n, con un alfabeto de tamaño a, siendo la distancia de edición $d = 1$, habrá n eliminaciones de caracteres posibles logrando que los términos que tienen que ser chequeados en el diccionario sean n.

Este algoritmo logra ser independiente del idioma ya que la cantidad de caracteres que posea el alfabeto del idioma no afecta la cantidad de chequeos a realizar sobre el diccionario.

4.4.1.2. Heurísticas

Se implementaron cuatro heurísticas, las cuales fueron evaluadas en los errores ortográficos encontrados en el conjunto de muestreo de desarrollo. Se prosigue a describir cada una de ellas y a continuación se muestran los resultados obtenidos.

Expansión del diccionario

En todas las heurísticas se agrega un pequeño diccionario conteniendo errores que ocurren frecuentemente en el muestreo de desarrollo y que no son detectados correctamente por ninguno de los *spellcheckers*. Este diccionario consta de diez entradas: la gran mayoría corresponde a palabras donde no se utilizó el apóstrofo correctamente. En él podemos encontrar palabras como: “doent” (corregida como “doesn’t”), “dont” (corregida como “don’t”) y “shi” (corregida como “she”).

Tres *spellcheckers* con resolución por mayoría simple en las sugerencias

Se obtienen todas las sugerencias de los tres *spellcheckers* para la palabra a verificar. Se ordenan todas las sugerencias por cantidad de ocurrencias. Dado que utilizamos tres *spellcheckers*, una sugerencia puede aparecer un máximo de tres veces y un mínimo de una vez.

La primera palabra en esta lista ordenada de sugerencias es la corrección más probable y se devuelve también las siguientes cuatro palabras como sugerencias alternativas.

Tres *spellcheckers* con diccionario personalizado y resolución por mayoría simple en las sugerencias

Esta heurística es idéntica a la anterior con la excepción de que se instancian los tres *spellcheckers* con un diccionario personalizado que contiene las palabras que se espera que un estudiante de inglés de nivel pre-A1/A1/A2 en el marco CEFR (Common European Framework of Reference for Languages), utilice según la lista de palabras de Cambridge Assessment English [5].

Tres *spellcheckers* con resolución mediante BERT

En esta heurística utilizamos BERT (sección 2.4.2) como un modelo de lenguaje para determinar la mejor sugerencia.

Se obtienen todas las sugerencias de los tres *spellcheckers* para la palabra a verificar. Luego se ejecuta BERT con la oración completa, donde se enmascara la palabra a verificar. De esta manera BERT utiliza el contexto, o las palabras que rodean a la palabra enmascarada para determinar cuáles son las palabras que tienen la mayor probabilidad de ocupar ese lugar en la oración. Se obtiene dicha probabilidad para cada una de las sugerencias obtenidas y se devuelve la sugerencia más probable como la corrección, y las cuatro más probables como sugerencias alternativas.

A continuación se describe un ejemplo a modo de ilustración. Si la oración es "She ir Andrea." y evaluamos la palabra "ir" para verificar si existe un error ortográfico, entonces se obtienen las sugerencias de los *spellcheckers*, que supongamos son las palabras "is", "if" e "I".

Se ejecuta BERT utilizando la oración "She <MASK> Andrea." donde la palabra "ir" se reemplaza por la máscara "<MASK>" para obtener las palabras con mayor probabilidad de ocupar esa posición en la oración. Luego evaluamos para cada sugerencia la probabilidad de que dicha sugerencia sea la palabra enmascarada:

$$P(\text{<MASK> == "is"}) = 0{,}0069$$
$$P(\text{<MASK> == "if"}) = 1{,}413\text{e-}06$$
$$P(\text{<MASK> == "I"}) = 1{,}032\text{e-}05$$

En este ejemplo, la palabra "ir" sería corregida utilizando la palabra "is" dado que fue la sugerencia con mayor probabilidad de ocupar ese lugar en la oración según el modelo de lenguaje.

Utilizando solamente Symspell

Esta heurística busca también mejorar la velocidad de corrección. Se utiliza solamente el *spellchecker* Symspell, el cual es el más veloz de los tres utilizados. Se busca observar

[5] `https://www.cambridgeenglish.org/Images/506166-starters-movers-flyers-word-list-2018.pdf` - Último acceso: Junio 2020

también si existe una gran diferencia al utilizar múltiples *spellcheckers*, ya que esta heurística es la única que utiliza uno solo.

Al igual que en las otras heurísticas, se devuelve la primera sugerencia como la corrección y las restantes cuatro (en caso de existir) como sugerencias alternativas.

4.4.1.3. Resultados

Para comparar las distintas heurísticas implementadas, se creó un conjunto de datos que consta de todas las palabras con errores ortográficos del muestreo de desarrollo y su correspondiente corrección esperada. Este conjunto consta de 66 palabras diferentes. Las heurísticas implementadas son evaluadas sobre este conjunto para observar el nivel de acierto de cada una.

Heurística	**Acierto(%)**
Tres *spellcheckers* con resolución por mayoría simple en las sugerencias	84
Tres *spellcheckers* con diccionario personalizado y resolución por mayoría simple en las sugerencias	71
Tres *spellcheckers* con resolución mediante BERT	74
Symspell	89

Tabla 4.2: Acierto en la corrección de palabras con errores de las distintas heurísticas

De los resultados presentes en la tabla 4.2 podemos ver que la heurística que utiliza un diccionario personalizado con la lista de palabras del nivel CEFR pre-A1/A1/A2 es la que obtiene peores resultados. La lista de palabras no logra cubrir el rango de palabras utilizadas por los estudiantes, haciendo que los *spellcheckers* elijan correcciones incorrectas y marquen como errores palabras que no lo son.

La heurística que utiliza un modelo de lenguaje (BERT), no logra elegir con efectividad la mejor sugerencia de los *spellcheckers*, eligiendo sugerencias en muchas ocasiones que resultan en oraciones no gramaticales. Otra desventaja de este método, la cual no se ve reflejada en las medidas de acierto, es que es un método notoriamente más lento que el resto de los explorados.

Entre el resto de las heurísticas, la que logra el mejor resultado, y que es además la más veloz, es la que utiliza solamente la librería *Symspell*.

Se ejecutaron las heurísticas también sobre un conjunto de palabras sin errores ortográficos para verificar la existencia de falsos positivos. La única heurística que falla esta prueba es la que utiliza un diccionario personalizado. En este caso, el diccionario encuentra errores ortográficos donde no los hay simplemente porque las palabras no se encuentran en el diccionario.

4.4.1.4. Corrección

La estrategia de corrección para este tipo de error implica cambiar la palabra con error utilizando la primer palabra de la lista de sugerencias. Por ejemplo, para la oración “I lke

orange juice.", la heurística encontrará las siguientes sugerencias para la palabra "lke":

"like", "lake", "lie", "the", "be"

Al elegir la primer sugerencia, la corrección queda de la siguiente manera:

*I **lke** orange juice.* → *I **like** orange juice.*

4.4.1.5. Ejemplo

Luego de pasar por el componente del pipeline de *spellchecker*, un texto como el que se muestra en la figura 4.2, quedará marcado como se muestra en la figura 4.3.

Andrea is wering a pinck t-shirt and shorts.

Figura 4.2: Entrada a detección de errores ortográficos

Andrea is <E_S orig="wering" sug="['wearing', 'during', 'spring', 'bring', 'ring']"> *wearing* </E_S> *a* <E_S orig="pinck" sug="['pink', 'since', 'piece', 'sick', 'kick']"> *pink* </E_S> *t-shirt and shorts.*

Figura 4.3: Salida luego de detección de errores ortográficos

4.4.2. Omisión de mayúsculas

Dentro de la omisión de mayúsculas se distinguen 3 casos:

- Al inicio de la oración.
- En el pronombre "I".
- En los nombres propios.

Una vez que el texto está segmentado en oraciones, se verifica que la primera palabra de cada oración comience con mayúscula. En caso contrario, se encierra dicha palabra con el tag **E_C-BOS** y en el parámetro "corr" se muestra la palabra mencionada comenzando con mayúscula.

El pronombre "I" es el pronombre personal en primera persona del singular. Se utiliza para que el hablante se refiera a sí mismo y se escribe con mayúscula por convención del lenguaje, aunque otros pronombres, como "he" o "she", no se escriben con mayúscula. Para verificar esto, se analiza cada oración buscando la palabra "i". En caso de encontrarse, se encierra la palabra "i" con el tag **E_C-PI** y en "corr" se setea "I".

Para el último caso se utiliza la funcionalidad de *Named Entities* de *Spacy* donde se detectan entidades como personas, nacionalidades, países y lenguajes, entre otros,

utilizando técnicas mencionadas en la sección 2.1.2.4. Al estar utilizando *Spacy* con modelos entrenados sobre textos en inglés, la funcionalidad *Named Entities* logrará identificar entidades en inglés y para contemplar el uso de nombres en español en los textos a procesar, se construyó un diccionario de nombres en español. El ejercicio resuelto por los estudiantes (descrito en el capítulo 3) indica que el nombre de la chica en la imagen es "Andrea". Este nombre es un nombre común tanto en inglés como en español, por lo que *Named Entities* logra identificarlo sin problemas. En el texto con identificador 21230 un estudiante decide nombrar al perro de "Andrea", y decide llamarlo "Paco", este nombre no es identificado por *Named Entities* como un nombre propio, pero es identificado como uno en el diccionario. Más allá del caso particular del perro "Paco", al complementar utilizando el diccionario, podremos identificar nombres en español en futuros ejercicios que se busquen corregir.

El diccionario de nombres propios en español fue construído a partir de datos recogidos del Instituto Nacional de Estadística de España[6]. Estos datos corresponden a la categoría: "todos los nombres con frecuencia mayor o igual a 20 personas" y poseen más de 50000 nombres de hombres y mujeres extraídos de la "Estadística del Padrón Continuo a la fecha 01/01/2020". En las entradas originales existían muchos nombres compuestos (como por ejemplo "José Manuel María"), estas entradas fueron "particionadas" en entradas de una palabra: "José", "Manuel" y "María". Al quedarnos con estos nombres, removiendo nombres con un largo menor a tres (correspondientes en su gran mayoría a nombres de origen asiático y de medio oriente) y eliminando repeticiones, el diccionario queda compuesto por 14272 nombres de hombres y mujeres.

Luego de reconocer estas entidades en el texto, se verifica que comiencen con mayúscula. De no ser así, se encierra la entidad en cuestión con el **tag E_C-PN** y en "corr" se muestra el nombre propio comenzando con mayúscula.

Se realizaron experimentos sobre el conjunto de muestreo de desarrollo para verificar si el diccionario de nombres identificaba alguna palabra como nombre propio, cuando en realidad la palabra era una palabra con error de ortografía, comprobándose que ninguna palabra en el muestreo de desarrollo mostró este comportamiento.

Si el nombre propio o el pronombre "I" con omisión de mayúscula se encuentra al comienzo de la oración, se omite marcar con el tag correspondiente pues el error ya fue detectado y marcado con el tag **E_C-BOS**.

4.4.2.1. Corrección

La estrategia de corrección para este tipo de error implica en los tres casos (inicio de oración (**E_C-BOS**), pronombre "I" (**E_C-PI**) y nombre propio (**E_C-PN**) cambiar a mayúsculas la primer letra de la palabra marcada.

Por ejemplo:

***she** likes to paint.* → ***She** likes to paint.*

[6] `https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736177009&menu=resultados&idp=1254734710990` - Último acceso: Noviembre 2020

4.4.2.2. Ejemplo

Para un texto como el que se muestra en la figura 4.4, el detector de errores del tipo *Omisión de mayúsculas* devolverá la salida mostrada en la figura 4.5.

she likes cycling with her pet dog carlos.
Carlos also likes when i go with them.

Figura 4.4: Entrada a detección de omisión de mayúsuculas

<E_C-BOS corr="She"> *she* </E_C-BOS> *likes cycling with her pet dog* <E_C-PN corr="Carlos"> *carlos* </E_C-PN> .
Carlos also likes when <E_C-PI corr="I"> *i* </E_C-PI> *go with them.*

Figura 4.5: Salida luego de detección de omisión de mayúsuculas

4.4.3. Concordancia sujeto-verbo

La concordancia sujeto-verbo se refiere a la relación que existe entre el sujeto y el predicado de una oración. Tiene que existir una concordancia entre el número y persona del sujeto y el verbo.

Por ejemplo, en la oración «She love her new kite» el sujeto es «She», el cual se encuentra en la tercera persona del singular, y el verbo es «love». La tercera persona del singular del verbo infinitivo «to love» es «loves». Por lo tanto, al no existir concordancia entre el número y persona del sujeto y el verbo, tenemos un error de concordancia sujeto-verbo en la oración.

Para detectar este tipo de error en los textos, se realizan dos tareas:

1. Identificar el sujeto y el verbo en la oración.
2. Verificar la concordancia entre el sujeto y el verbo.

4.4.3.1. Identificar el sujeto y el verbo en la oración

Para realizar esta tarea se investigaron dos técnicas distintas, utilizando un parser de dependencias y un método basado en diccionarios.

Parser de dependencias

El árbol resultante de aplicar un parser de dependencias sobre una oración tiene la información necesaria para determinar el sujeto y el verbo. El arco *nsubj* nos dirige al sujeto mientras que el verbo es la raíz del árbol.

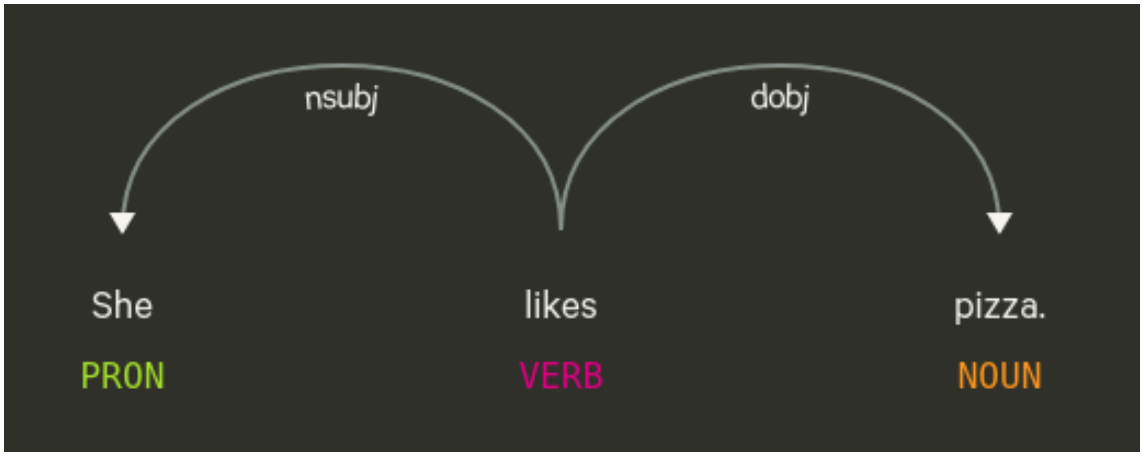


Figura 4.6: Ejemplo de árbol resultado al utilizar un parser de dependencias sobre la oración “She likes pizza.”

Este enfoque logra identificar el sujeto y el verbo de la oración cuando los textos son correctos o poseen una pequeña cantidad de errores (o ruido) como pueden ser los textos de los estudiantes que obtuvieron un puntaje de cinco o seis.

Cuando se ejecuta el parser de dependencias sobre las oraciones de los textos con puntajes menores a cinco, podemos ver que el árbol resultado no siempre logra identificar el sujeto y el verbo de la oración de manera correcta.

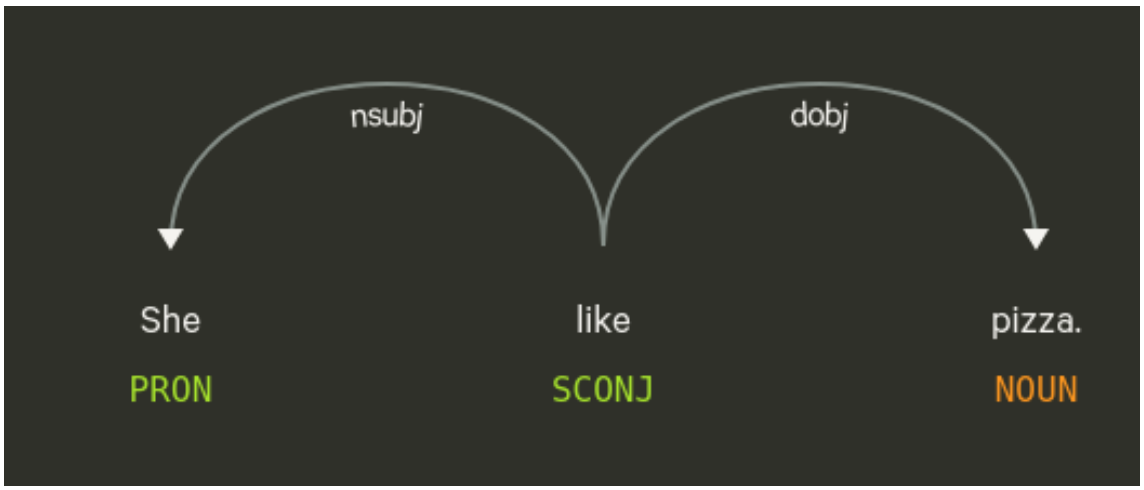


Figura 4.7: Ejemplo de árbol resultado al utilizar un parser de dependencias sobre la oración “She like pizza.”

En la figura 4.6 el parser de dependencias identifica de manera correcta al sujeto “She” y al verbo “likes”. Mientras que en la figura 4.7 el error de concordancia que estamos tratando de detectar, entre las palabras “She” y “like”, genera un árbol resultado donde el parser de dependencias confunde al verbo “like” con una conjunción de subordinación (SCONJ).

El uso de las etiquetas asignadas por el etiquetador morfológico *POS tagger* no logra

solucionar el problema. Las etiquetas asignadas a oraciones agramaticales no siempre logran capturar al sujeto y al verbo.

Método basado en diccionarios

Al observar la estructura de las oraciones escritas por los estudiantes en el muestreo de desarrollo, se encontró que casi en su totalidad, el sujeto es un pronombre personal o un nombre propio, independientemente del puntaje obtenido por el estudiante.

Utilizamos este dato para buscar específicamente pronombres personales o nombres propios entre el comienzo de la oración y la ubicación del verbo. Para identificar el verbo, compilamos una lista con 1000 verbos extraídos del *Oxford Advanced Learner's Dictionary*[18] donde nos aseguramos de incluir todos los verbos presentes en *Oxford 5000 wordlist* [29] que proporciona una lista de 5000 palabras curada por Oxford la cual contiene las palabras consideradas más importantes para los estudiantes de inglés, basado en su frecuencia y relevancia en el rango completo A1-C2 del CEFR (Common European Framework of Reference for Languages).

En caso de no encontrar un sujeto (pronombre personal o nombre propio) se busca un sustantivo entre el comienzo de la oración y el verbo. El sustantivo es asignado entonces como el sujeto de la oración.

4.4.3.2. Verificar la concordancia entre el sujeto y el verbo

Una vez identificados el sujeto y el verbo de la oración, resta verificar si éstos concuerdan en número y persona. Para realizar esta tarea se intentaron dos enfoques: uno basado en un modelo de lenguaje (BERT) y otro basado en reglas. Dentro del enfoque basado en reglas, se investigó la adición de reglas al parser (*mal-rules*) y un método que utiliza lexicones para verificar el cumplimiento de las reglas de manera más directa.

Verificar la concordancia utilizando BERT

Al utilizar un modelo de lenguaje para verificar la concordancia nos basamos en la siguiente hipótesis: al poner una máscara sobre el verbo de la oración, indicarle al modelo que prediga las probabilidades de las palabras que pueden reemplazar la máscara, y verificar cuál de las distintas combinaciones número/persona del verbo original obtiene una mayor probabilidad. El modelo asignará la mayor probabilidad al verbo que concuerde en número/persona con el sujeto de la oración, dado que es esa versión del verbo la que produce una oración correcta en inglés. Podemos luego comparar el verbo más probable con el verbo del texto original para determinar si hubo o no un error de concordancia. Este enfoque se inspira en [13] donde se utiliza esta técnica para evaluar las capacidades sintácticas de BERT.

Se tomará como ejemplo la siguiente oración:

«Andrea like to eat pizza.»

Una vez identificados el sujeto «Andrea» y el verbo «like», preparamos la oración para que sea procesada por BERT. Para ello enmascaramos el verbo, obteniendo la siguiente oración:

«Andrea <MASK> to eat pizza.»

Una vez procesada, el modelo asigna probabilidades a las palabras que podrían reemplazar la máscara. En este caso solo nos interesa verificar las probabilidades de «like» y «likes».

$$P(\text{“like”}) = 0{,}000298$$
$$P(\text{“likes”}) = 0{,}001582$$

El modelo predice que «likes» es más probable. Siguiendo nuestra hipótesis, «likes» será entonces la palabra que genere una oración sin error de concordancia en este caso.

El siguiente paso consiste en verificar si la palabra original es la misma que la palabra elegida por el modelo. En este caso, al no serlo, se marca el error de concordancia.

Verificar la concordancia utilizando reglas

Este enfoque busca crear reglas que identifiquen estructuras en las oraciones que produzcan errores de concordancia.

Las reglas implementadas no buscan identificar todas las formas de errores de concordancia que pueden existir en una oración, sino aquellas que fueron observadas en el muestreo de desarrollo y que son esperados en la producción gramatical de un estudiante de nivel A1/A2 como se describe en el apéndice A.

Algunas de las reglas creadas podrían ser unificadas para lograr una menor cantidad en total. Pero se decidió crear las reglas de forma que cada una determine un error específico, por lo que si se quiere realizar una corrección, o generar una devolución sobre el error, se posea la información necesaria para hacerlo.

Regla 1

Esta regla identifica los casos donde el sujeto es un pronombre personal en tercera persona del singular o un nombre propio y donde el verbo no se encuentra en tercera persona. Ejemplos de este error se encuentran en oraciones como:

[She]$_{\text{S}}$ [sing]$_{\text{V}}$.
[Juana]$_{\text{S}}$ [run]$_{\text{V}}$ around the park every afternoon.

Regla 2

Esta regla identifica los casos donde el sujeto no es un pronombre personal en tercera persona del singular y donde el verbo se encuentra en tercera persona del singular. Ejemplos de este error se encuentran en oraciones como:

[I]$_{\text{S}}$ [eats]$_{\text{V}}$.
[We]$_{\text{S}}$ [sings]$_{\text{V}}$ our favorite songs.

Regla 3

Esta regla identifica los casos donde no debería haber concordancia entre el sujeto y el verbo cuando aparece entre ellos el verbo auxiliar «does». Para esta regla buscamos el caso donde el sujeto es el pronombre personal «she», «he», o un nombre propio. Ejemplos de este error se encuentran en oraciones como:

[She]$_{\text{S}}$ does [likes]$_{\text{V}}$ to play.
[She]$_{\text{S}}$ doesn't [eats]$_{\text{V}}$ fast food.

Regla 4

Esta regla identifica los casos donde no hay concordancia entre el sujeto y el verbo cuando aparece entre ellos el verbo auxiliar «do». Para esta regla buscamos el caso donde el sujeto es el pronombre personal «I», «we», «you» o «they». Ejemplos de este error se encuentran en oraciones como:

[I]$_{\text{S}}$ do [likes]$_{\text{V}}$ to play.
[We]$_{\text{S}}$ don't [drinks]$_{\text{V}}$ enough water.

Regla 5

Esta regla identifica los casos donde no hay concordancia entre el sujeto y el verbo cuando aparece entre ellos el verbo modal «can». Ejemplos de este error se encuentran en oraciones como:

[I]$_{\text{S}}$ can [runs]$_{\text{V}}$ very fast.
[We]$_{\text{S}}$ can't [drinks]$_{\text{V}}$ coffee.

Cabe notar que las reglas identifican también errores donde el verbo auxiliar, específicamente las variantes del infinitivo «to do», no concuerda en número/persona con el sujeto. Dada la prevalencia de errores de concordancia entre el sujeto y el verbo auxiliar en el muestreo de desarrollo, se incorporaron reglas para verificar estos casos.

Este conjunto de cinco reglas, se inspira en la idea de *mal-rules* [30], en donde se busca extender la gramática de forma de identificar estructuras que posean errores. Si bien no

estamos extendiendo ninguna gramática en este caso, utilizamos dos métodos diferentes para identificar las reglas: uno que utiliza información sintáctica y morfológica proveniente del *pipeline* de *spacy*, y otro que busca identificar los distintos elementos presentes en las reglas mediante el uso de lexicones.

Reglas mediante el uso de información sintáctica y morfológica

Para implementar este método se utiliza la funcionalidad *Matcher*[7] perteneciente a la librería *spacy*. Esta funcionalidad permite declarar reglas para identificar frases o palabras utilizando los atributos extraídos a lo largo de los diferentes pasos del *pipeline* de *spacy*.

Podemos utilizar, entonces, para definir las reglas, información como el texto exacto del *token* que buscamos, su etiqueta *POS*, la etiqueta asignada por el parser de dependencia o por el componente *NER*. Se puede verificar también si el *token* fue identificado como correspondiente a puntuación, espacio, número o comienzo de oración, entre otros. La web de *spacy* proporciona una herramienta web[8] para explorar y crear reglas, pudiendo realizar cambios y ver los efectos de las reglas en diferentes textos de forma interactiva.

Reglas mediante el uso de lexicones

En este método, se utilizó un enfoque inspirado en [37], donde se utilizan lexicones (o diccionarios) de verbos junto con un *POS-tagger* y reglas para determinar la concordancia entre el sujeto y verbo de las oraciones.

Antes de verificar las reglas descritas anteriormente, se separan los sujetos y los verbos en dos categorías: tercera persona del singular y no-tercera persona del singular.

Para los sujetos, en el primer grupo tenemos los pronombres «she», «he» e «it», junto con los nombres propios y los sustantivos singulares. En el grupo no-tercera persona del singular, tenemos los pronombres «I», «you», «we» y «they», junto con los sustantivos plurales.

En cuanto a los verbos, se utilizan los 1000 verbos obtenidos de *Oxford Advanced Learner's Dictionary*[18] y *Oxford 5000 wordlist*[29]. Se crean dos diccionarios, uno con los verbos en tercera persona del singular y uno con los verbos en la no-tercera persona del singular. Un detalle importante es que la intersección de estos diccionarios de verbos es vacía: no existe un verbo que pertenezca a los dos diccionarios a la vez, por lo que al encontrar un verbo en un diccionario, sabremos con certeza si este se encuentra en la tercera persona del singular o no. Para crear el diccionario de los verbos en la tercera persona del singular, se siguieron las siguientes pautas:

- La mayor parte de los verbos en inglés agregan una "-s" al final de la forma base del verbo para crear la tercera persona del singular ("love**s**", "run**s**", "hope**s**").
- Los verbos que terminan en "-ch", "-s", "-sh", "-x", o "-z" lo hacen agregando "-es". ("watch**es**", "miss**es**", "mix**es**").

[7] `https://spacy.io/api/matcher` - Último acceso: Agosto 2021

[8] `https://spacy.io/universe/project/matcher-explorer` - Último acceso: Agosto 2021

- Los verbos que terminan en consonante seguido de “y”, lo hacen sustituyendo “y” por “i” y agregando “es”, como el caso de “try” y “tr**ies**”.
- Los verbos que terminan en “o” típicamente agregan “-es”. Dado que existen solamente cinco verbos en el diccionario que terminan en “o”, se agregó de manera manual su tercera persona. Estos verbos son: “go”→“goes”, “boo”→“boos”, “outdo”→“outdoes”, “undergo”→“undergoes”, “forgo”→“forgoes” y “undo”→“undoes”.
- Los verbos irregulares no siguen estas reglas y tienen formas especiales. Estos verbos fueron agregados de forma manual: “do”→“does”, “has”→“have”, “say”→“says” y “to be”→“is”.

Con estos recursos, podemos identificar el número y persona del verbo, y de los sujetos que son pronombres personales. Para identificar el número y persona de los sustantivos, nombres propios y la existencia de verbos auxiliares o negaciones en la oración, se utiliza el *POS tagger*. Los verbos auxiliares en inglés se diferencian de los verbos principales, ya que deben utilizarse como parte de algunos tiempos verbales. Ellos son “be”, “have”, “do” y “will”. También están los verbos auxiliares modales; los principales son “shall”, “would”, “could”, “can”, “should”, “might” y “must”.

Una vez determinados todos los elementos, se verifica si alguna de las reglas definidas se cumple, y de cumplirse, se marca el error de concordancia.

A modo de ejemplo, se aplicará el método descripto anteriormente a la siguiente oración:

«She doesn’t likes to eat apples.»

La primera tarea involucra identificar al sujeto y al verbo principal en la oración. Se encuentra el *token* “She” al comienzo de la oración, el cual identificamos como un pronombre personal, en tercera persona del singular. Dado que los pronombres personales en inglés pertenecen a una clase cerrada (sección 2.1.2.2), podemos identificarlos de forma directa en el texto.

Para identificar al verbo principal, se busca identificar al primer verbo que sucede al sujeto en la oración. En este paso se ignoran los verbos auxiliares, los más comunes en los textos son “can” y “do”. El verbo “do” (“does”) en esta oración actúa como verbo auxiliar. Dado que la oración es una negación, es necesario utilizar “do” como verbo auxiliar para construir la oración. Este uso del verbo se conoce como *do-support* [8] y se utiliza para construir negaciones y preguntas en inglés.

Se identifica “likes” entonces como el verbo, y se verifica a cuál diccionario de verbos pertenece para determinar su número. En este punto sabemos que el verbo se encuentra en tercera persona del singular.

Tenemos ahora los elementos necesarios para verificar si alguna de las cinco reglas antes definidas se cumple. Encontramos al sujeto “She” (tercera persona del singular), encontramos al verbo “likes” (tercera persona del singular) y encontramos el *token* “does” entre ellos. La regla número tres abarca este caso, en esta oración existe, entonces, un error de concordancia ya que el verbo debería ser “like”.

4.4.3.3. Resultados

A modo de resumen, para detectar este tipo de error se identificaron dos tareas. Para la primera, identificar el sujeto y el verbo en la oración, se exploraron dos técnicas diferentes. La técnica basada en utilizar un parser de dependencias no logra identificar el sujeto y el verbo con certeza. En (Nagata et al., 2011)[24] se explora cómo algunas herramientas de PLN obtienen peores resultados al ser utilizadas sobre textos ESL (*English as a Second Language*).

El segundo método de identificación, basado en diccionarios, logró cumplir la tarea, por lo que fue elegido para trabajar en conjunto con los métodos que verifican la concordancia (segunda tarea).

En la segunda tarea, se investigaron tres métodos para verificar la concordancia. Uno de ellos, buscaba explorar el uso de *mal-rules* utilizando la información sintáctica y morfológica disponible para intentar encontrar estructuras gramaticales erróneas. Este método no logra verificar la concordancia, encontrando problemas similares a los encontrados al utilizar el parser de dependencias en la primera tarea.

Para los restantes dos métodos, evaluamos los métodos utilizando los textos del muestreo de desarrollo en donde se encontraron 42 errores de concordancia sujeto-verbo y los resultados se muestran en la tabla 4.3.

Heurística	**Precision**	**Recall**	**F1-Score**
Concordancia utilizando BERT	0.77	0.73	0.75
Concordancia utilizando reglas - versión lexicones	0.82	0.76	0.79

Tabla 4.3: Performance de las mejores heurísticas para identificar errores de concordancia sujeto-verbo en el muestreo de desarrollo

4.4.3.4. Corrección

La estrategia de corrección para este tipo de error implica cambiar la conjugación del verbo (ya sea el verbo principal, o el verbo auxiliar "do") para que cumpla la concordancia con el sujeto. Al tener la heurística implementada como un conjunto de reglas, cuando se detecta un error sabemos la conjugación que debe tener el verbo para cumplir la concordancia. Usando esta información junto con la librería *Lemminflect*[9], la cual nos permite obtener las diferentes conjugaciones del verbo, podemos obtener la conjugación correcta.

Por ejemplo:

*She doesn't **likes** to eat pizza.* $\rightarrow$ *She doesn't **like** to eat pizza.*

[9] `https://pypi.org/project/lemminflect/`

4.4.3.5. Ejemplo

Para un texto como el que se muestra en la figura 4.8, el detector de errores del tipo **SVA** devolverá la salida mostrada en la figura 4.9.

She doesn't likes to eat apples.

Figura 4.8: Entrada a detección de concordancia sujeto-verbo.

She doesn't <E_SVA corr="like"> *likes* </E_SVA> *to eat apples.*

Figura 4.9: Salida luego de detección de concordancia sujeto-verbo.

4.4.4. Forma del verbo

Los errores de forma verbal pueden dividirse en distintas categorías. Podemos distinguir entre ellas errores en el uso del tiempo verbal, errores en el uso de verbos irregulares, errores en el uso del infinitivo y el gerundio, y la concordancia entre el sujeto y el verbo.

Luego de analizar el muestreo de desarrollo, encontramos que la mayor parte de los errores relacionados a la forma verbal cometidos por los estudiantes son de concordancia entre el sujeto y el verbo, y en el uso del infinitivo y el gerundio.

Los errores de concordancia fueron analizados en la sección 4.4.3 y tienen su tratamiento específico, por lo que en esta sección se analizarán los errores en el uso del infinitivo y el gerundio.

La confusión entre el uso del infinitivo y el gerundio es un error común entre los estudiantes de inglés dado que en ciertas situaciones es correcto utilizar cualquiera de las dos formas, aunque existen situaciones donde utilizar una forma y no la otra es incorrecto.

Forma base	**Infinitivo**	**Gerundio**
walk	to walk	walking
take	to take	taking
swim	to swim	swimming

Tabla 4.4: Ejemplos de infinitivo y gerundio.

En la tabla 4.4 se muestran algunos ejemplos de infinitivo y gerundio junto con el verbo en su forma base. El infinitivo en el idioma inglés se construye al agregar "to" antes de la forma base del verbo, mientras que el gerundio se construye en la mayor parte agregando el sufijo "-ing" a la forma base del verbo. Las reglas para formar el resto de los gerundios son:

- Si la forma base del verbo termina en "-e", se omite la última letra y se agrega "-ing" para construir el gerundio. Por ejemplo: tak**e** → tak**ing**, slid**e** → slid**ing** y rid**e** → rid**ing**.
- Si la forma base del verbo termina en "-ie", se reemplaza "ie" por "y" y se agrega "-ing". Por ejemplo: d**ie** → d**ying**, t**ie** → t**ying** y l**ie** → l**ying**.

- Si la última sílaba de la forma base del verbo tiene la forma consonante-vocal-consonante y es la sílaba tónica, se construye el gerundio duplicando la última letra y agregando "-ing". Por ejemplo: s**wim** → swim**ming**, be**gin** → begin**ning** y **tap** → tap**ping**.

Al analizar los errores de este tipo en el muestreo de desarrollo, encontramos que la omisión del "to" al formar el infinitivo domina las ocurrencias. Un 84 % (48 de 57 errores encontrados) corresponde a este tipo de error. Tres ejemplos de textos pertenecientes al muestreo de desarrollo con distintos puntajes donde podemos ver este tipo de error son:

Extraído del texto con ID 13184 y puntaje 4:

She likes read and go to shool → *She likes* ***to*** *read and* ***to*** *go to shool*

Extraído de texto con ID 56585 y puntaje 3:

She doesn't like sing. → *She doesn't like* ***to*** *sing.*

Extraído de texto con ID 60359 y puntaje 2:

... i like study i dont like fishing →... i like ***to*** *study i dont like fishing*

En estos casos "to" no actúa como una preposición sino como parte del infinitivo y debe estar presente para que la oración sea gramaticalmente correcta.

Los restantes nueve errores encontrados en el muestreo de desarrollo corresponden a errores donde no se respeta el paralelismo cuando existe más de un verbo en gerundio o infinitivo.

El concepto de paralelismo en este contexto refiere al uso de estructuras gramaticales idénticas dentro de la misma oración o frase. En este caso el uso consistente del infinitivo o gerundio. Un ejemplo de una oración que no respeta el paralelismo y su correspondiente corrección sería:

We like singing, ***to draw*** *and writing.* → *We like singing,* ***drawing*** *and writing.*

A la hora de detectar esta regla, debemos tener en cuenta dos casos especiales. El primero, cuando encontramos el verbo *to go*, el cual es infinitivo pero puede ser continuado con verbos en gerundio. Por ejemplo en la oración: *She likes to go running and to draw.*. El segundo caso sucede cuando tenemos una lista de verbos en infinitivo, en esta situación el primer verbo debe poseer "to", pero el resto de los verbos no tiene por qué. Por ejemplo en la oración: *They love to run, dance and jump.*

Además de detectar los casos observados en el muestreo de desarrollo, se busca detectar también los casos descriptos a continuación, los cuales fueron recopilados de [32].

Adjetivos antes del infinitivo

Si la palabra inmediatamente antes del infinitivo o gerundio es un adjetivo, entonces la forma correcta es utilizar el infinitivo.

It's not [easy]$_{ADJ}$ [to speak]$_{INF}$ English.

Existen casos como *"It's nice meeting you."* donde la regla no aplica, pero incluso en estos casos, la oración *"It's nice to meet you."* es también correcta, comunica la misma intención y es más frecuente en el inglés escrito (mientras que la primera es más frecuente en el inglés oral).

Infinitivo luego de pronombres personales acusativos o sustantivos

Si la palabra inmediatamente antes del infinitivo o gerundio es un pronombre personal acusativo, entonces la forma correcta es utilizar el infinitivo.

We want [her]$_{PPA}$ [to sing]$_{INF}$ tonight.

De igual manera, si en lugar de un pronombre personal acusativo, tenemos un sustantivo, debemos utilizar el infinitivo.

I ask the [teacher]$_{SUST}$ [to help]$_{INF}$ me.

Gerundio luego de preposiciones

Si la palabra inmediatamente antes del infinitivo o gerundio es una preposición, entonces la forma correcta es utilizar el gerundio.

They talk [about]$_{PREP}$ [going]$_{GER}$ to the party today.

Esta regla tiene como excepciones a las preposiciones "but" y "except", donde podemos encontrar oraciones como *"I have no choice but to follow them."* las cuales deben utilizar el infinitivo.

Un caso interesante sucede aquí con la preposición "to". Por ejemplo en frases verbales como "*to look forward to*" y "*to get used to*", el último "to" es una preposición, y siguiendo esta regla, se debe utilizar un gerundio. Como por ejemplo en la oración:

She didn't really take [to]$_{PREP}$ [studying]$_{GER}$ French.

Este caso particular donde se tienen frases verbales con la preposición "to" son excluídos de la detección. Esta decisión se debe a la complejidad de este tipo de construcciones, junto con que no se observaron casos de este tipo o similares en el conjunto

de desarrollo y a la dificultad a la hora de detectar si el *token* "to" pertenece al infinitivo o si es una preposición. Agregar una regla para detectar este caso puede resultar en una regla muy específica y que puede llevar a generar más falsos positivos que detecciones correctas.

Verbos que requieren una forma específica

Existen verbos que siempre introducen un infinitivo y verbos que siempre introducen un gerundio, mientras que también existen verbos con los cuales es posible utilizar ambas formas (aunque utilizar una o la otra puede cambiar el significado de la oración).

Algunos verbos que requieren el uso del infinitivo son: "agree", "hope", "learn" y "want".

He wants ***to draw*** *tonight.*

They hope ***to see*** *you soon.*

Mientras que algunos verbos que requieren el uso del gerundio son: "admit", "advise", "recommend" y "suggest".

I suggest ***singing*** *as a social activity.*

I recommend ***running*** *as one of the best forms of exercise.*

Entre los verbos que pueden utilizarse con el infinitivo y el gerundio encontramos: *like*, *hate*, *love* y *propose*.

I like ***to swim****.* $\longleftrightarrow$ *I like* ***swimming****.*

She proposes ***to pay*** *for the trip.* $\longleftrightarrow$ *She proposes* ***paying*** *for the trip.*

4.4.4.1. Método implementado para la detección

El procedimiento implementado para identificar este tipo de error involucra el uso de diccionarios de verbos y un etiquetador morfológico.

Se crearon dos diccionarios de verbos, uno con aquellos verbos que introducen al uso del infinitivo y otro con los verbos que introducen el uso del gerundio.

El método procesa una oración a la vez. La definición de oración en este caso es un poco laxa, dado que los textos no siempre se encuentran correctamente puntuados, y existen casos donde múltiples oraciones son procesadas a la vez.

Se comienza identificando los distintos pares de sujeto y verbo, utilizando la técnica descrita en la sección 4.4.3.1 (identificar al sujeto y verbo utilizando diccionarios) ya que

habíamos encontrado que es más efectiva a la hora de identificar los pares sujeto-verbo que las herramientas sintácticas y morfológicas sobre estos textos.

Una vez detectados el sujeto y el verbo principal se busca identificar a los verbos que se encuentren en la forma base o en gerundio de forma de identificar unidades que tengan la siguiente forma:

$$\text{... (sujeto) ... (verbo principal) ... } (v_1) \text{ ... } (v_2) \text{ ... } (v_n) \text{ ...}$$

Una vez encontradas todas las unidades con sus correspondientes elementos de anclaje, pasamos a verificar las distintas reglas antes descritas.

Para evaluar las reglas, ubicamos al *token* correspondiente a v_1, el cual supongamos que se encuentra en la posición i de la oración. Llamemos *predecesor*, al *token* inmediatamente anterior a v_1, el *token* con posición $i - 1$. Si este *token* es "to", se actualiza *predecesor* al *token* anterior (posición $i - 2$).

Se continúa moviendo *predecesor* a la izquierda hasta encontrar un *token* que sea una preposición, un adjetivo, un sustantivo, un pronombre personal acusativo o el verbo principal identificado en la unidad.

Una vez determinado *predecesor*, podemos determinar que regla verificar y si v_1 se encuentra en la forma verbal adecuada.

En caso de que exista más de un v_n, se verifica que se cumpla el paralelismo donde todos los v_i poseen la misma forma verbal.

Método aplicado a un ejemplo.

Apliquemos este método sobre una oración de ejemplo. Supongamos que tenemos la siguiente oración:

She likes eat pizza, walk at night and singing.

El primer paso es identificar a la unidad (sujeto)-(verbo principal)-(v_n). En esta oración existe una sola unidad y la identificamos como:

[She]$_{\mathbf{S}}$ [likes]$_{\mathbf{V}}$ [eat]$_{\mathbf{v1}}$ pizza, [walk]$_{\mathbf{v2}}$ at night and [singing]$_{\mathbf{v3}}$.

El siguiente paso es identificar a *predecesor* a partir de v_1 (en este caso "eat"). Encontramos que *predecesor* es el verbo principal "likes", y a través del uso de diccionarios determinamos que este verbo principal puede introducir tanto infinitivos como gerundios.

En estos casos se decide elegir la forma que posea v_1, en este caso al estar en la forma base, buscaremos que los verbos se encuentren en infinitivo.

El verbo v_1 ("eat") no se encuentra en la forma verbal correcta ya que no posee "to" para formar el infinitivo. Aquí se detecta un error.

Al haber más de un verbo v_n, debemos verificar que se cumpla la regla del paralelismo. Debemos verificar que el resto de los verbos se encuentre en infinitivo.

El verbo v_2 ("walk") se encuentra en infinitivo de manera correcta, si bien no posee "to", habíamos mencionado un caso especial en el paralelismo en el cual es correcto obviar "to" cuando tenemos una lista de verbos en infinitivo.

El verbo v_3 ("singing") se encuentra en gerundio, por lo que no cumple la regla del paralelismo. Aquí se detecta otro error de forma verbal.

Al finalizar encontramos dos errores, el primero en *eat* $\rightarrow$ *to eat* y el segundo en *singing* $\rightarrow$ *sing*.

4.4.4.2. Corrección

La estrategia de corrección para este tipo de error depende del subtipo de error encontrado. Podemos encontrar dos categorías de error: la primera donde debemos cambiar la forma del verbo de la forma base al gerundio (y viceversa), y la segunda donde debemos agregar o remover la partícula "to" que forma el infintivo.

Para la primera categoría, dada la implementación en reglas de la heurística, sabemos la conjugación correcta que debe tener el verbo por lo que utilizamos nuevamente la librería *Lemminflect* para obtenerla.

En la segunda categoría, si debemos remover la partícula "to", simplemente identificamos donde se encuentra y la removemos. Y en el caso que debamos agregarla para formar el infinitivo, la agregamos inmediatamente antes del verbo. Aquí tomamos en cuenta que no encontramos occurrencias de *split-infinitive*[8] en el muestreo de desarrollo. El *split-infinitive* sucede cuando se utilizan adverbios o frases adverbiales entre la partícula "to" y el verbo. Un ejemplo famoso de este tipo de construcción forma parte de la introducción del programa televisivo *Star Trek*:

***To** boldly **go** where no man has gone before.*

Ejemplos de las correcciones para los errores de forma verbal son:

*She enjoys to read and **drawing**.* $\rightarrow$ *She enjoys to read and **draw**.*

She likes swim. $\rightarrow$ *She likes **to** swim.*

4.4.4.3. Ejemplo

Para un texto como el que se muestra en la figura 4.10, el detector de errores del tipo **VF** devolverá la salida mostrada en la figura 4.11.

She likes <E_VF> *paint* </E_VF>*, eat and sing.*

Figura 4.11: Salida luego de detección de forma del verbo.

She likes paint, eat and sing.

Figura 4.10: Entrada a detección de forma del verbo.

4.4.5. Uso de determinantes

En gramática, el determinante es un término que se usa antes de un sustantivo para mostrar a qué instancia particular de la clase denotada por el sustantivo se está haciendo referencia. En las frases "my first boyfriend" y "that strange woman", las palabras "my" y "that" son determinantes. [10]

Se reconocen distintos tipos de determinantes:

- **Artículos:** the, a, an
- **Demostrativos:** that, this, these, those
- **Palabras de diferencia:** other, another
- **Distributivos:** each, every, either, neither
- **Números:** one, thirty, fifty-three
- **Posesivos:** my, your, his, her, its, our, their
- **Nombres propios posesivos:** Bill's, Andrea's
- **Cuantificadores:** a little, a few, a lot of, all, any, both, enough, many, most, much, some

En cuanto al uso de determinantes en los ejemplos del muestreo de desarrollo, se reconocieron dos tipos de errores: omisión o uso innecesario de éstos. Se puso mayor énfasis en la corrección de determinantes de tipo artículos y posesivos ya que son los más utilizados por niños con este nivel de inglés.

Como parte de la corrección del tipo de error *Missing article* donde se debe incluir el artículo "a" o "an" antes del sustantivo o antes de los adjetivos en caso de que hubiera, se agrega un estudio del sonido de la primer letra del sustantivo en cuestión. Si el sonido es vocálico, se sugiere la corrección "an" y si el sonido es consonántico se sugiere "a".

A continuación se muestra un ejemplo donde se omite el artículo para el sustantivo "bike". Este tipo de error fue marcado con el tag **E_MA** (*Missing article*) y en el parámetro "corr" se asigna "a" ya que la primer letra de "bike" tiene sonido consonántico.

She can ride bike.

She can ride *<E_MA corr="a">* *bike* *</E_MA>*.

[10] `https://dictionary.cambridge.org/es/diccionario/ingles/determiner` - Último acceso: Diciembre 2020

Por otro lado, se detectó el uso innecesario de artículos como en el siguiente ejemplo. Este tipo de error fue marcado con el tag **E_UA** (*Unnecessary article*).

She likes reading a books.

She likes reading a <E_UA> *books* </E_UA>.

Para detectar este tipo de error se utiliza el etiquetador POS de *Spacy*. En cada oración se detectan los sustantivos. Si el sustantivo es plural y en la posición anterior inmediata hay un determinante, se detecta el error y el sustantivo se encierra mediante el tag **E_UA**. También se tiene en cuenta si en las posiciones anteriores al sustantivo se encuentran adjetivos describiéndolo. Por ejemplo:

She is wearing a white trousers.

She is wearing a white <E_UA> *trousers* </E_UA>.

Si el sustantivo es singular, se verifica si el sustantivo es contable o no. Para ello se utilizó una lista de sustantivos adecuados al nivel de inglés de los textos donde se especifica la distinción de conteo. Si el sustantivo es contable se verifica la existencia de un determinante en la posición anterior inmediata; de lo contrario se detecta el error encerrando el sustantivo mediante el tag **E_MA**. En el ejemplo a continuación el sustantivo "bike" es singular y contable por lo tanto le hace falta un artículo:

She is riding bike.

She is riding <E_MA> *bike* </E_MA>.

Si es no contable, no debería estar a continuación de un determinante. De ser así, se encierra sólo el sustantivo con el tag **E_UA**. Por ejemplo en este caso "hair" es no contable por lo tanto el artículo "a" es innecesario:

She has a blond hair.

She has a blond <E_UA> *hair* </E_UA>.

4.4.5.1. Corrección

La estrategia de corrección al encontrar errores donde existe un uso innecesario del artículo (**E_UA**) implica simplemente remover el artículo correspondiente al sustantivo marcado.

Un ejemplo de este tipo de corrección es:

She has ***a*** *beautiful pets.* $\rightarrow$ *She has beautiful pets.*

La corrección cuando se detecta un error de omisión de artículo (**E_MA**) implica determinar el artículo a utilizar y la ubicación del mismo.

Para determinar la ubicación del artículo es necesario verificar si existen adjetivos modificando al sustantivo marcado. En caso de existir adjetivos y conjunciones, el artículo debe ser incorporado de forma que los adjetivos se encuentren entre el sustantivo y el artículo.

En cuanto a la elección del artículo, los artículos indefinidos "a" y "an" son determinados de acuerdo a como "suenan" los sustantivos (o adjetivo más cercano en caso de haberlo). La regla general dice que utilizaremos "an" cuando el sustantivo es singular y comience con el sonido de una vocal (o con una "h" muda). Mientras que utilizaremos "a" cuando el sustantivo es singular y comience con el sonido de una consonante (o una vocal que "suene" como una consonante)[32].

Para determinar si una palabra comienza con un sonido de vocal o consonante no alcanza con saber si la palabra comienza con una letra vocal o consonante. Por ejemplo la palabra "university", comienza con la vocal "u", pero esta vocal "suena" como "y" al pronunciar la palabra, por lo que debemos utilizar el artículo indefinido "a".

Utilizamos el diccionario de pronunciación *CMU Pronouncing Dictionary*[11] el cual posee la información necesaria para determinar si el primer fonema de una palabra corresponde a una vocal o a una consonante.

En el caso en donde la palabra marcada no exista en el diccionario de pronunciación, agregamos "a/an", indicando que en ese lugar debe haber un artículo aunque no podamos determinar con exactitud cual.

Un ejemplo de este tipo de corrección es:

She has unique set of interests. $\rightarrow$ *She has* ***a*** *unique set of interests*

4.5. Ejemplo

En esta sección se mostrará el funcionamiento paso a paso del corrector a través de un texto de puntaje 5 que se muestra en la figura 4.12.

[11] `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

Andrea is fourteen years old , she have a little cute dog, sometimes she goes riding with her dog , she is wearing a pink t-shirt, white shorts and white trainers,her bike have a basket and have a darck red colour.
Andrea loves eating pizza but she hate apples, she loves reading books and she is very good at maths, i think she is an un-healthy girl because she eats a lot and she hates apples. But she is very clever

Figura 4.12: Entrada del corrector

El primer paso en el *pipline* del corrector consiste en el preprocesamiento del texto. La salida de este primer componente se muestra en la figura 4.13. En este caso no existen caracteres extraños por lo que no se perciben cambios luego de la limpieza del texto. A continuación se separa el texto en oraciones según el signo de puntuación ".".

Andrea is fourteen years old , she have a little cute dog, sometimes she goes riding with her dog , she is wearing a pink t-shirt, white shorts and white trainers, her bike have a basket and have a darck red colour.

Andrea loves eating pizza but she hate apples, she loves reading books and she is very good at maths, i think she is an un-healthy girl because she eats a lot and she hates apples.

But she is very clever

Figura 4.13: Texto luego del preprocesamiento

Luego de preprocesado el texto, se pasa a detectar y corregir los errores de ortografía. Aquí se detectan dos errores de ortografía en las palabras "darck" y "un-healthy". En la figura 4.14 se muestra el texto luego de pasar por el corrector ortográfico.

Andrea is fourteen years old, she have a little cute dog, sometimes she goes riding with her dog, she is wearing a pink t-shirt, white shorts and white trainers, her bike have a basket and have a <E_S orig="darck" sug="['dark', 'back', 'park', 'mark', 'dance']"> *dark* </E_S> *red colour.*

Andrea loves eating pizza but she hate apples, she loves reading books and she is very good at maths, i think she is an <E_S orig="un-healthy" sug="['unhealthy']"> *unhealthy* </E_S> *girl because she eats a lot and she hates apples.*

But she is very clever

Figura 4.14: Texto luego de pasar por el corrector ortográfico

Una vez que se han corregido los errores ortográficos, se está en condiciones

de realizar el procesamiento del texto en los otros 4 componentes: *Omisión de mayúsculas*, *Manejo de artículos*, *Concordancia sujeto-verbo* y *Forma verbal*. En cada componente se trabaja oración por oración buscando errores de cada tipo. El texto marcado, que forma parte de la salida del corrector para el ejemplo presentado, es el que se muestra en la figura 4.15 donde resaltados con colores se pueden ver los tags para cada tipo de error.

Andrea is fourteen years old, she <E_SVA corr="has"> *have* </E_SVA> *a little cute dog, sometimes she goes riding with her dog, she is wearing a pink t-shirt, white shorts and white trainers, her bike have a basket and have a* <E_S orig="darck" sug="['dark', 'back', 'park', 'mark', 'dance']"> *dark* </E_S> *red colour.*

Andrea loves eating pizza but she <E_SVA corr="hates"> *hate* </E_SVA> *apples, she loves reading books and she is very good at maths,* <E_C-PI corr="I"> *i* </E_C-PI> *think she is an* <E_S orig="un-healthy" sug="['unhealthy']"> *unhealthy* </E_S> *girl because she eats a lot and she hates apples.*

But she is very clever

Figura 4.15: Texto marcado

Aquí se puede observar que en la primera sentencia, que a su vez está compuesta por varias oraciones separadas por coma, existen dos errores de concordancia que no fueron detectados en el verbo "have". Esto se debe a que el sujeto "her bike" no fue detectado por el identificador de sujetos.

Como parte del posprocesamiento, se integra el texto en una sola unidad tal como estaba al comienzo del procesamiento, pero con los tags de marcado. A continuación, se procede a corregir el texto según las correcciones que incluyen los tags. La salida del corrector incluye el texto marcado con tags y el texto corregido. En la figura 4.16 se muestra el texto corregido.

Andrea is fourteen years old, she has a little cute dog, sometimes she goes riding with her dog, she is wearing a pink t-shirt, white shorts and white trainers, her bike have a basket and have a dark red colour.
Andrea loves eating pizza but she hates apples, she loves reading books and she is very good at maths, I think she is an unhealthy girl because she eats a lot and she hates apples.
But she is very clever

Figura 4.16: Texto corregido

En el apéndice B se muestran varios ejemplos con su correspondiente salida del corrector y además, un análisis de los errores detectados.

4.6. Resultados en muestreos

Los resultados obtenidos para cada tipo de error al corregir el muestreo de desarrollo se muestran en la tabla 4.5. Se puede apreciar que la detección de la falta de mayúsculas al comienzo de la sentencia y en el pronombre "I" son los que arrojan mejores resultados. En cuanto a las mayúsculas en nombres propios, se obtuvo un buen valor en *recall* pero se introdujeron varios falsos positivos en palabras como "apple" que es reconocida como organización cuando en realidad se hace referencia a la fruta.

El peor resultado se da en la detección de errores en el uso de artículos, en especial cuando el uso del artículo "the" es innecesario. Por ejemplo, para la oración "She likes the pizza" el uso del artículo "the" no es necesario ya que está haciendo referencia al gusto por la pizza en general, pero como es gramaticalmente correcto nuestro detector no lo marca. Otro ejemplo en el que ocurre lo mismo es en la oración "The books are better than music" donde el artículo "the" es innecesario ya que se está hablando de los libros en general.

Por otro lado, se introducen varios falsos positivos en el tipo de error *Artículo faltante* en algunos casos donde el *POS-tagger* de *Spacy* etiqueta adjetivos como sustantivos como en "the sports routine" o en "computer lessons".

Error	**Desarrollo**			**Evaluación**		
	Pre	**Rec**	**F**	**Pre**	**Rec**	**F**
Mayúscula en pronombre "I"	1.0	1.0	1.0	-	-	-
Mayúscula al comienzo de la oración	0.99	1.0	0.99	0.92	0.79	0.85
Ortografía	0.89	0.88	0.88	0.81	0.85	0.83
Mayúscula en nombres propios	0.73	1.0	0.84	0.75	1.0	0.86
Forma del verbo	0.73	0.91	0.81	0.66	0.81	0.72
Concordancia sujeto-verbo	0.82	0.76	0.79	0.83	0.77	0.80
Artículo faltante	0.71	0.87	0.78	0.50	0.81	0.62
Artículo innecesario	0.67	0.67	0.67	0.38	0.75	0.5

Tabla 4.5: Métricas para cada tipo de error en ambos conjuntos

Luego de dejar una versión estable del corrector, se procedió a evaluarlo corrigiendo los textos del muestreo de evaluación y compararlos con los corregidos a mano por los tutores del proyecto. En la tabla 4.5 también se muestran estos resultados.

En este subconjunto de datos no se encontraron casos para el tipo de error *Mayúscula en pronombre "I"*. Para la mayoría de los tipos de error restantes, se mantienen las métricas semejante a las obtenidas en el muestreo de desarrollo.

En el tipo de error *Artículo innecesario* se encontraron nuevos casos donde el artículo innecesario no se encuentra previo a un sustantivo (ya se plural o singular) sino previo a un verbo, y este tipo de error no había sido contemplado. Además se introducen varios falsos positivos en oraciones donde los adjetivos son etiquetados como sustantivos como ya se mencionó anteriormente. Esto lleva a que, por ejemplo,

en el sintagma “a fishing rod”, “fishing” es etiquetado como un sustantivo no contable haciendo referencia al deporte y donde el artículo “a” es innecesario, cuando en realidad está describiendo al sustantivo “rod”.

Capítulo 5

Clasificador

5.1. Idea general

Una vez finalizada la implementación del corrector de textos, surgió la necesidad de validarlo. Una forma de validar las heurísticas de corrección sería aplicarlas a un escenario de corrección automática. Para eso se utilizaron las heurísticas de todos los textos del corpus corregidos; en la sección 5.2 se muestran algunos resultados sobre la detección de errores sobre el corpus entero.

Existe un antecedente, descrito en la sección 5.3, que realizó este experimento donde se entrenó un clasificador sobre el mismo conjunto de datos utilizando *Bag of Words* como *features*. Por lo tanto, se procedió a replicar el uso de la técnica que proporcionó mejores resultados en el experimento previo (utilizando el mismo conjunto de entrenamiento y validación) para luego comparar los resultados obtenidos. Posteriormente se entrenó un clasificador a partir de los mismos conjuntos, pero esta vez utilizando como *features* las heurísticas encontradas por el corrector automático.

5.2. Detección de errores en corpus

Luego de desarrollado y testeado el corrector automático de errores, se procedió a marcar todos los textos del corpus. A través de este proceso se obtuvo una planilla con el texto original, el texto marcado y la cantidad de errores por tipo de error de cada texto. En la figura 5.1 se muestra el promedio de la cantidad de errores por texto para cada puntaje.

	Spell	C-BOS	C-PI	C-PN	SVA	UA	MA	VF
0	4.43	1.08	0.02	0.20	0.16	0.02	0.16	0.09
1	2.08	1.22	0.09	0.19	0.27	0.03	0.42	0.30
2	1.53	1.50	0.10	0.21	0.62	0.06	0.53	0.84
3	1.22	1.68	0.03	0.19	0.86	0.11	0.60	1.48
4	1.33	1.52	0.01	0.15	0.91	0.18	0.68	1.70
5	1.76	1.36	0.01	0.10	1.33	0.14	0.87	1.58
6	2.08	1.38	0	0.08	0.62	0.23	1.0	0.54

Tabla 5.1: Promedio de cantidad de errores por texto para cada puntaje

Analizando la tabla 5.1 se puede observar que los errores de ortografía (**Spell**) son más frecuentes en los textos con puntaje más altos (5 y 6) y los más bajos (0 y 1). Esto puede explicarse por la gran cantidad de palabras en español presentes en los textos de puntaje más bajo y por la mayor producción escrita en los textos calificados con mayor puntaje dando lugar a más errores ortográficos.

Para el resto de los tipos de error, el promedio se mantiene casi constante al variar el puntaje. Para los casos de concordancia sujeto-verbo (**SVA**), artículo faltante (**MA**) y forma del verbo (**VF**) se puede observar un leve aumento del promedio de errores por texto a medida que aumenta el puntaje.

5.3. Trabajo previo

En esta sección se hará un resumen del reporte de estudio guiado realizado en el grupo de PLN [16] y que fue utilizado como punto de partida para validar las heurísticas de corrección del presente trabajo. Dicho estudio fue realizado sobre el mismo corpus de textos escritos por estudiantes escolares en el contexto de la prueba adaptativa descripta en la sección 3. Una aclaración pertinente es que este antecedente consideró también las otras partes del ejercicio de *Writing* donde el estudiante debía completar con palabras las oraciones dadas, mientras que el corrector observa solamente la parte de producción escrita del ejercicio.

Utilizando *Bag of Words* (BoW) como *features* se compararon los resultados al probar distintos algoritmos de aprendizaje automático: *Naive Bayes* (NB), Perceptrón multicapa (MLP), máquinas de vector soporte (SVM) y modelo de máxima entropía (ME, también conocido como Regresión Logística). También se investigó estos métodos utilizando distintas dimensiones de BoW y n-gramas.

Una estrategia explorada en el reporte consiste en la agrupación de puntajes en franjas. Mediante esta estrategia se busca reducir la disparidad en la cantidad de ejemplos en cada clase a predecir. Las franjas fueron definidas de la siguiente manera:

- **Bajo:** comprendiendo las pruebas con puntaje 0 y 1
- **Medio:** comprendiendo las pruebas con puntaje 2 y 3

- **Alto:** comprendiendo las pruebas con puntaje 4, 5 y 6.

La tabla 5.2 muestra los mejores resultados de acierto y macro-F obtenidos para cada método junto con la cantidad de palabras utilizadas en la BoW y la estructura de los n-gramas (unigramas solamente, unigramas y bigramas, bigramas y trigramas, etc) utilizados para alcanzar estos resultados sobre el conjunto de validación. Estos modelos fueron entrenados para predecir el puntaje de un texto en el rango original de cero a seis.

Método	**Acierto**	**Macro-F**	**Palabras**	**N-gramas**
NB	45.44 %	23.18 %	5000	2-3
SVM	27.79 %	14.34 %	500	2-2
MLP	62.63 %	41.37 %	1000	1-3
ME	67.05 %	44.93 %	500	1-2

Tabla 5.2: Acierto por método para el modelo de datos original (obtenida de [16])

La tabla 5.3 muestra los mejores resultados de acierto y macro-F obtenidos para cada método, pero esta vez al entrenar los modelos para predecir la franja (baja, media, alta) a la cual corresponden los puntajes.

Método	**Acierto**	**Macro-F**	**Palabras**	**N-gramas**
NB	66.42 %	49.94 %	3000	3-3
SVM	42.95 %	32.00 %	750	1-3
MLP	83.28 %	70.05 %	750	1-2
ME	86.11 %	73.00 %	750	1-2

Tabla 5.3: Acierto por método para el modelo de datos por franja (obtenida de [16])

En el reporte interno de trabajo dirigido, al evaluar los resultados de las tablas 5.2 y 5.3, el modelo de máxima entropía (ME) con BoW de 750 palabras (utilizando unigramas y bigramas) fue elegido y los resultados de clasificar el conjunto de test los podemos ver en la tabla 5.4.

Acierto	0.8553
macro F1	0.7061
F1 Bajo	0.8907
F1 Medio	0.8276
F1 Alto	0.4000

Tabla 5.4: Resultado al clasificar utilizando ME con 750 palabras sobre el conjunto de test (obtenida de [16])

5.4. Partición del conjunto de datos

Con el objetivo de comparar los resultados obtenidos en el reporte, realizamos la misma partición en conjuntos de entrenamiento (70 % del total), validación (15 % del total) y test (15 % del total). Las particiones no serán idénticas a las utilizadas en el reporte dado que desconocemos la composición exacta de las mismas. Pero el reporte especifica la cantidad de textos por puntaje en las particiones y utilizamos estas proporciones para construir nuestras particiones.

La totalidad de los textos fueron procesados utilizando el corrector automático. Los textos vacíos (11.667), los textos no vacíos sin nota asignada (24), los textos conteniendo sólo números (104), los textos conteniendo caracteres no alfanuméricos (5) y un texto que arrojó error al ser procesado por el corrector fueron descartados a la hora de realizar las particiones.

	Reporte			**Nuestro**		
Nota	**Entrenam.**	**Validación**	**Test**	**Entrenam.**	**Validación**	**Test**
0	16101	3451	3450	9464	2028	2029
1	9123	1956	1955	7971	1708	1708
2	12368	2651	2650	11928	2556	2556
3	7206	1545	1544	7197	1542	1542
4	945	203	202	945	202	203
5	94	21	20	94	21	20
6	9	2	2	9	2	2
Total	**45846**	**9829**	**9823**	**37608**	**8059**	**8060**

Tabla 5.5: Distribución por puntaje de los conjuntos de entrenamiento, validación y test.

En la tabla 5.5 se muestra la distribución por puntaje de un total de 53.727 textos sobre los distintos conjuntos (entrenamiento, validación y test). Bajo la columna *Reporte* se muestra la distribución por puntaje de los distintos conjuntos utilizados en el reporte, mientras que bajo la columna *Nuestro* se encuentra la distribución de textos por puntajes que utilizaremos para realizar nuestros experimentos.

Nuestros conjuntos poseen un menor número de textos comparados con aquellos utilizados en el reporte, esto es debido a los textos descartados por los motivos mencionados anteriormente. La mayor parte de los textos descartados corresponden a aquellas clases que se encuentran sobre-representadas, observando que para los puntajes 4, 5 y 6, la cantidad de textos es idéntica tanto en los conjuntos utilizados en el reporte como en los conjuntos que utilizaremos aquí.

Como se había mencionado, en el reporte se optó por agrupar los puntajes en franjas para combatir la disparidad de textos pertenecientes a las diferentes clases. En el presente trabajo se replicó la misma idea.

En la tabla 5.6 podemos ver la distribución de los textos en las clases definidas como franjas.

Al utilizar los puntajes en el rango [0-6], se puede observar que la relación entre la cantidad de textos pertenecientes a la clase con más elementos (puntaje cero:

	Reporte			Nuestro		
Franja	**Entrenam.**	**Validación**	**Test**	**Entrenam.**	**Validación**	**Test**
Baja	25224	5407	5405	17435	3736	3737
Media	19574	4196	4194	19125	4098	4098
Alta	1048	226	226	1048	225	225
Total	**45846**	**9829**	**9823**	**37608**	**8059**	**8060**

Tabla 5.6: Distribución por franjas de los conjuntos de entrenamiento, validación y test.

$16101 + 3451 + 3450 = 23002$ textos) y la clase con menos elementos (puntaje seis: $9 + 2 + 2 = 13$ textos) es aproximadamente de 1769 a 1. Mientras que al agrupar en franjas, esta relación entre la clase "Baja" $(25224 + 5407 + 5405 = 36036)$ y la clase "Alta" $(1048 + 226 + 226 = 1500)$ es aproximadamente de 24 a 1.

5.5. Modelos explorados

Una vez obtenidas las particiones en conjuntos de entrenamiento, validación y test, buscaremos reproducir en primer lugar los resultados obtenidos en el reporte utilizando el método que obtuvo mejores resultados. Utilizaremos también la cantidad de errores (de distinto tipo) detectados por el corrector automático como *features* y en última instancia entrenaremos un modelo utilizando BoW y la cantidad de errores como *features* comparando finalmente los resultados obtenidos por los diferentes modelos.

5.5.1. Modelos utilizando BoW de palabras como *features*

En el reporte se experimentó con cuatro métodos de aprendizaje automático: *Naive Bayes*, *Support Vector Machines*, perceptrón multicapa y modelo de máxima entropía. El método que alcanzó los mejores resultados fue el modelo de máxima entropía utilizando como *features* un BoW de los 750 unigramas y bigramas más frecuentes por lo cual será el método que replicaremos.

En la figura 5.1 podemos observar los 10 unigramas y bigramas más frecuentes del corpus. Se puede ver aquí la dependencia que tendrá el modelo a la pregunta sobre la cual se basan las respuestas, ya que unigramas como "pizza", "andrea" y "14" son específicos de este contexto y no serán útiles para predecir el puntaje de otro ejercicio. Dicho de otra forma, si se propone un examen con otro ejercicio donde, por ejemplo, los niños tienen que describir la rutina de una persona llamada Carlos y de 10 años de edad, estas palabras probablemente no quieran decir nada.

```
('she', 134977)
('is', 78095)
('and', 74214)
('like', 65127)
('she is', 42814)
('pizza', 40702)
('andrea', 40481)
('old', 33212)
('14', 32707)
('years', 32454)
```

Figura 5.1: 10 unigramas y bigramas más frecuentes junto con su cantidad de ocurrencias.

Siguiendo la metodología del reporte, y dado que estamos buscando replicar los resultados obtenidos por un método específico (en lugar de buscar el mejor método a utilizar) utilizamos el conjunto de entrenamiento junto con el conjunto de validación para entrenar el modelo, y luego calcular las métricas utilizando las predicciones sobre el conjunto de test. Incluímos también el uso de un método que no fue explorado en el reporte: *Random Forest*.

	Por puntaje		**Por franja**	
	Acierto	Macro-F1	Acierto	Macro-F1
ME-reporte	-	-	0,85	0,71
ME	0,62	0,40	0,83	0,70
RF	0,67	0,48	0,86	0,74

Tabla 5.7: Resultados para modelos con BoW

En la tabla 5.7 los resultados obtenidos por el modelo ME del reporte y los obtenidos por nuestro ME son similares, teniendo una diferencia de 2 % en el acierto y 1 % en macro-F1. Esta comparación solamente se puede realizar utilizando el modelo de datos por franjas dado que en el reporte no se calculan estas métricas para el modelo de datos por puntaje (utilizando un conjunto de entrenamiento compuesto por la concatenación del conjunto de entrenamiento y validación). Podemos ver también que el método *Random Forest* logra resultados similares al ME, manteniendo una diferencia de entre 3 %-4 % aproximadamente en las métricas al compararlo con los modelos ME explorados.

Al lograr reproducir los valores alcanzados en las métricas con el mejor método encontrado en el reporte podemos establecer una línea base para comparar con modelos que utilicen los errores encontrados por el corrector automático como *features*.

5.5.2. Modelos utilizando errores como *features*

Utilizando las mismas particiones de entrenamiento, validación y test entrenaremos ahora modelos con un conjunto de *features* diferente. Utilizaremos los errores detectados por el corrector automático para predecir el puntaje/franja de un texto. Los nueve *features* utilizados se muestran en la tabla 5.8.

Feature	Descripción
E_spell	Cantidad de errores de ortografía.
E_c-bos	Cantidad de errores de omisión de mayúscula al comienzo de la oración.
E_c-pi	Cantidad de errores de omisión de mayúscula en el pronombre "I".
E_c-pn	Cantidad de errores de omisión de mayúscula en nombres propios.
E_vf	Cantidad de errores de forma del verbo.
E_sva	Cantidad de errores de concordancia sujeto-verbo.
E_ma	Cantidad de errores de falta de artículo.
E_ua	Cantidad de errores de uso innecesario de artículo.
length	Largo del texto en palabras.

Tabla 5.8: Definición de *features* utilizados.

El último *feature*, "length", fue agregado ya que al observar los textos producidos por los estudiantes se puede apreciar que textos más largos (en cantidad de palabras) tienden a corresponder con puntajes más altos. Puede entonces que exista información relevante representada en este *feature*, la cual ayude a tomar una decisión más acertada al modelo.

	Por puntaje		**Por franja**	
	Acierto	Macro-F1	Acierto	Macro-F1
NB	0,48	0,32	0,73	0,61
RF	0,56	0,37	0,79	0,64
ME	0,59	0,35	0,82	0,64
SVM	0,59	0,36	0,82	0,63
MLP	0,60	0,37	0,82	0,68

Tabla 5.9: Resultados de modelos utilizando conjunto de features de errores.

Los resultados al entrenar diferentes modelos utilizando el conjunto de entrenamiento y evaluando sobre el conjunto de validación se muestran en la tabla 5.9. Podemos ver que MLP es el que logra mejores resultados en este caso.

5.5.3. Modelos utilizando *features* combinados

Por último, a modo de experimento, se entrenaron los modelos combinando los dos conjuntos de *features* explorados. Este conjunto de *features* estará compuesto entonces por los 8 tipos de errores detectados por el corrector automático, el largo en palabras del texto a predecir, y los 750 unigramas y bigramas más frecuentes.

Los resultados al entrenar los diferentes modelos utilizando la partición de entrenamiento y evaluando sobre la partición de validación se encuentran en la tabla 5.10. El método *Random Forest* es el que alcanza mejores resultados al igual que cuando utilizamos BoW como conjunto de *features*.

	Puntaje		Franja	
	Acierto	Macro-F1	Acierto	Macro-F1
MLP	0,32	0,08	0,51	0,23
SVM	0,33	0,12	0,51	0,22
NB	0,44	0,29	0,70	0,61
ME	0,63	0,41	0,84	0,71
RF	0,68	0,49	0,86	0,76

Tabla 5.10: Resultados de los modelos utilizando *features* combinados.

5.6. Resultados generales

En las secciones anteriores encontramos el mejor método para los distintos conjuntos de *features*. Estos son: el modelo de *Random Forest* (**RF**) sobre el conjunto de *features* **BoW**, el modelo de perceptrón multicapa (**MLP**) sobre el conjunto de *features* **Errores+length** y nuevamente el modelo de *Random Forest* (**RF**) sobre el conjunto de *features* **Combinados**.

Ahora para comparar los resultados a los obtenidos en el reporte, seguimos la misma metodología allí utilizada y entrenamos utilizando un conjunto de entrenamiento compuesto por la unión de los conjuntos de entrenamiento y validación, y evaluaremos sobre el conjunto de test. Los resultados utilizando los distintos conjuntos de *features* y modelos de datos (puntaje/franjas) se encuentran en la tabla 5.11[1].

	Puntaje		Franja	
	Acierto	Macro-F1	Acierto	Macro-F1
BoW-reporte (ME)	-	-	0.85	0.70
BoW (RF)	0.67	0.47	0.86	0.74
Errores+length (MLP)	0.60	0.36	0.83	0.65
Combinados (RF)	0.68	0.48	0.86	0.76

Tabla 5.11: Resultados al evaluar los mejores modelos de cada conjunto de *features* sobre la partición de test.

Lo primero a observar de los resultados, es que al compararlos con los resultados obtenidos en las secciones anteriores, podemos decir que los distintos modelos no parecen sufrir de *overfitting* dado que los resultados obtenidos al ser evaluados sobre la partición de validación y la partición de test son similares ($\approx 2\,\%$ de diferencia en promedio). Por lo que los modelos logran generalizar relativamente bien las predicciones sin "memorizar" ejemplos vistos durante el entrenamiento.

El modelo que utiliza BoW es el que logra mejores resultados al ser comparado con el que utiliza los tipos de errores ($+3\,\%$ en acierto y $+9\,\%$ en macro-F1 al comparar con las métricas obtenidas en el reporte), aunque este último tiene la ventaja de ser un modelo independiente de la pregunta sobre la cual los textos

[1]El reporte no realiza la evaluación utilizando la partición de test sobre el modelo de datos por puntaje.

fueron producidos. El modelo de BoW queda adaptado al ejercicio, y no es portable a otro ejercicio de manera sencilla. Es lógico que sus resultados serán mejores, porque se conocen las palabras correctas. Al utilizar los tipos de errores detectados como *features*, no se conocen las palabras frecuentes y aún así sus resultados se encuentran bastante cerca; esto nos motiva a tomar este modelo como base mejorable y así poder utilizarlo para clasificar textos que correspondan a diferentes preguntas sin apegarnos al contexto.

En cuanto al modelo que utiliza la unión de los conjuntos de *features* (**Combinados**), podemos ver que si bien logra mejorar los resultados obtenidos por el modelo (**Errores+length**), la combinación de los *features* no logra captar nueva información y los resultados son similares a aquellos obtenidos por el modelo (**BoW**).

También se hicieron pruebas quitando el *feature* del largo del texto para evaluar cuál es su influencia en los resultados del clasificador. Se observó que el largo del texto aporta aproximadamente el 10 % en los resultados del clasificador utilizando sólo los datos obtenidos a partir del detector de errores. De esta observación se puede concluir que el largo del texto aporta información interesante, pero no es definitorio al momento de clasificar un texto.

Continuando con los experimentos donde se excluyó el largo del texto dentro de los *features* para entrenar el clasificador combinado, se observó que dicho dato aporta casi la misma información que la bolsa de palabras más usadas. En la tabla 5.12 se muestran los resultados para poder hacer ésta comparación.

	Por puntaje		**Por franja**	
	Acierto	Macro-F1	Acierto	Macro-F1
Errores + *length*	0,59	0,33	0,82	0,63
Errores + BoW	0,60	0,38	0,83	0,67

Tabla 5.12: Resultados en testing combinando errores con *length* o con BoW

Finalmente, en las figuras 5.2 y 5.3 tenemos las matrices de confusión para los modelos que utilizan el conjunto de *features* **errores+length** y **BoW** (correspondiente al reporte) respectivamente.

El modelo **BoW** no posee ninguna instancia de clasificar un texto "Alta" como "Baja" o "Baja" como "Alta", mientras que el modelo **errores+length** posee una sola instancia donde clasifica un texto "Baja" como "Alta" pero ninguna instancia donde un texto "Alta" se clasifica como "Baja".

Un texto que pertenezca a la clase "Media", en caso de que sea clasificado de manera incorrecta (falso negativo), es más probable que sea confundido con la clase "Baja" independientemente del clasificador. Este sesgo puede deberse a la diferencia en la cantidad de elementos pertenecientes a cada clase en el conjunto de datos. Se experimentó con técnicas de *oversampling* y *undersampling* pero los resultados obtenidos durante la fase de validación de los modelos no mostraron una mejoría en los resultados por lo que se decidió continuar sin utilizar estas técnicas.

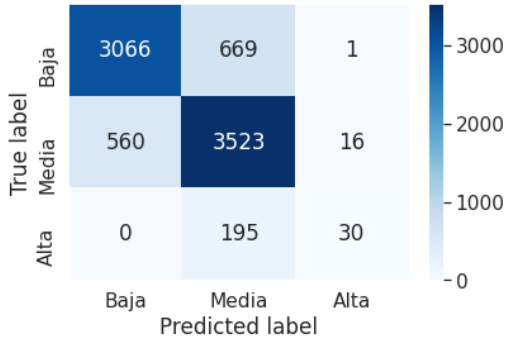


Figura 5.2: Modelo **errores+length** por franjas.

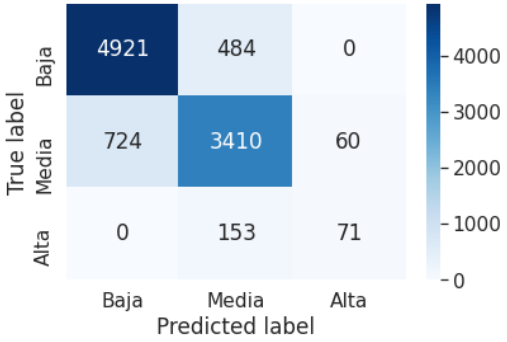


Figura 5.3: Modelo **BoW** por franjas (extraído del reporte).

También podemos ver en la matrices de confusión que el modelo **errores+length** no logra reconocer textos pertenecientes a la clase "Alta" de igual manera que el modelo **BoW**. Este último clasifica correctamente 71 de ellos mientras el primer modelo logra clasificar correctamente 30. Es posible que el modelo **BoW** logra capturar mejor los puntajes mayores porque captura nociones de cuáles palabras son más apropiadas para este ejercicio. Por otra parte, el modelo **errores+length** tiene menos falsos positivos al clasificar textos de la clase "Alta" (16) comparado con los falsos positivos para esta clase obtenidos por el modelo **BoW** (60).

Ambos modelos logran clasificar de manera correcta la mayor parte de los textos de las clases "Baja" y "Media". Y ambos modelos encuentran dificultades a la hora de clasificar textos de la clase "Alta" con la mayor parte de ellos siendo falsos negativos clasificados bajo la clase "Media", nuevamente mostrando la dificultad de los modelos, independientemente del conjunto de *features* utilizados, para clasificar textos pertenecientes a esta clase.

Capítulo 6

Conclusiones y trabajo a futuro

En esta sección se presentan las conclusiones obtenidas al realizar el presente proyecto. Se incluye, además, una descripción de los recursos generados a lo largo de su desarrollo y se presentan líneas de trabajo a futuro para mejorar lo hasta aquí trabajado.

6.1. Recursos generados

En primer lugar, se destaca la creación de un conjunto de heurísticas para evaluar y corregir textos escritos por niños. Dado un texto, esta herramienta detecta, marca y corrige los tipos de errores más frecuentes encontrados en niños de edad escolar que comienzan a acercarse a la lengua inglesa.

A partir del corpus de textos escritos por niños de Primaria con nivel inicial de Inglés facilitado por Ceibal, se agregó una columna donde se encuentra el texto original marcado con tags identificando los distintos errores detectados luego de procesado y una columna para cada tipo de error (8 en total) indicando la cantidad de errores de dicho tipo encontrados en ese texto para 54007 ejemplos. Este recurso podrá ser utilizado en trabajos posteriores sobre corrección automática de textos escritos por niños.

Por último, dentro de los recursos generados se encuentra un modelo que dado un texto escrito por un niño cuyo nivel de inglés es inicial, lo clasifica asignándole una categoría (Alta, Media o Baja) según la calidad de lo desarrollado en dicho texto.

6.2. Conclusiones

En este proyecto se trabajó en el desarrollo de métodos para la corrección automática de errores en textos cortos escritos por niños con un nivel inicial de inglés (A1/A2 CEFR). Para cumplir este objetivo, se plantearon metas más específicas al inicio del proyecto que fueron abordadas y alcanzadas.

Se estudiaron distintos artículos y técnicas utilizadas en el área del PLN - Corrección de Errores Gramaticales (GEC) relacionados con la corrección automática de textos escritos por estudiantes de inglés, comenzando con técnicas basadas en reglas y métodos estadísticos, técnicas neuronales utilizando aprendizaje automático, técnicas inspiradas en la tarea de traducción automática (*Machine Translation*) hasta el método que hoy en día alcanza el estado del arte (en la sección 2.2 - GECToR), el cual es un método de etiquetado secuencia a secuencia (*seq2seq*) que utiliza *transformers*.

Obtuvimos dos conjuntos de textos escritos por niños de Educación Primaria representativos del estilo de textos que se busca corregir. Denominamos a estos textos como muestreo de desarrollo y muestreo de validación. El muestreo de desarrollo fue utilizado para estudiar la naturaleza de los textos y definir los tipos de errores más frecuentes y sobre los cuales centrar el presente trabajo. El segundo muestreo sería utilizado luego para validar y comparar la calidad del corrector una vez finalizada su implementación. Estos textos provienen de un corpus con 65528 textos, el cual fue posteriormente procesado y etiquetado por el corrector automático.

Con el conjunto de errores a trabajar definido, se realizó la implementación de distintas técnicas para abordarlos y así poder generar heurísticas de corrección para cada uno. Para evaluar estas heurísticas se llevó a cabo el etiquetado a mano de todos los textos, el muestreo de desarrollo por parte nuestra, y el muestreo de validación por parte de los tutores del proyecto.

Los tipos de errores más frecuentes encontrados en ambos muestreos fueron en su mayor parte consistentes, aunque no completamente. Por ejemplo en el muestreo de desarrollo, se encontró que de forma frecuente los estudiantes no escribían el pronombre "I" en mayúscula, lo cual se comprende dado que no existe una regla de este estilo en el idioma español, pero este tipo de error no fue encontrado en ningún texto perteneciente al muestreo de validación. Por otra parte en el muestreo de validación, aparece el error *Missing subject* como uno de los más frecuentes, el cual no se encuentra entre los más frecuentes en el muestreo de desarrollo. La falta del sujeto o el verbo en una oración presenta un problema para el corrector, dado que estos elementos se utilizan como "puntos de anclaje" para poder procesar los errores de concordancia sujeto-verbo (SVA) y los errores de forma verbal (VF).

A la hora de elegir las herramientas y métodos a utilizar se tuvo en cuenta que el corrector tiene que ser construído dentro de la realidad del uso que se le pueda dar en el ámbito de la Educación Primaria. Esto implica tener presente la velocidad de procesamiento de los textos y de los recursos computacionales utilizados por el corrector. Se tomaron decisiones como por ejemplo no implementar la corrección de *real-word errors* (sección 2.3), dado que un método efectivo encontrado para la corrección implicaba utilizar BERT como modelo de lenguaje, y procesar a través de él cada palabra del texto. Incorporar BERT al proyecto con este propósito afectaría de manera adversa el tiempo de procesamiento. Otra razón por la cual se tomó esta decisión fue la poca frecuencia de este tipo particular de errores, lo que podría derivar en la incorporación de una gran cantidad de falsos positivos.

Fue interesante observar a la hora de desarrollar y explorar las diferentes heurísticas, la diferencia que existe en los resultados proporcionados por las

herramientas del *pipeline* de PLN de *spacy* (*parser*, etiquetador morfológico y NER) al tratar con los textos de puntajes altos y bajos. Las herramientas logran identificar de forma efectiva los diferentes componentes de las oraciones cuando se proporcionan textos de puntaje alto, aunque estos textos tengan errores (lo que en términos de las herramientas podemos considerar como ruido). En el caso de los textos con puntajes bajos, no siempre logran identificar los distintos componentes por lo que se optó por el uso de diccionarios, junto con la estructura de las oraciones que se busca procesar para identificar algunos elementos. Las herramientas sintácticas y morfológicas fueron de todas maneras utilizadas, pero para identificar elementos que no fueran ambiguos de manera frecuente (palabras pertenecientes a clases morfológicas cerradas y algunas clases abiertas como los adjetivos). Las soluciones basadas en reglas y diccionarios también tienen sus limitantes, no logran capturar todos los casos de error y pueden también sufrir problemas de ambigüedad. Una ventaja en este caso es que muchos de esos problemas se manifestarían en caso de intentar detectar errores sobre textos con construcciones gramaticales más complejas, mientras que la variedad de estructuras gramaticales y uso del léxico se encuentra de alguna forma acotado por el nivel de inglés de los estudiantes y el tipo de ejercicio a realizar.

Una vez finalizada la implementación de todas las heurísticas y el *pipeline* de corrección, se procesó el muestreo de validación y se comparó los resultados obtenidos entre ambos muestreos. Una preocupación a la hora de la implementación utilizando el muestreo de desarrollo fue el intentar no sobre-ajustar al mismo. Se busca crear un corrector que sea capaz de encontrar errores en cualquier texto de este estilo, más allá de los que utilizamos como referencia. Las medidas al evaluar indican que en gran medida se logró esto. Si bien las medidas tienden a ser más bajas, se identificaron en el muestreo de validación situaciones que no ocurrían en el muestreo de desarrollo. Estos nuevos casos deberían ser estudiados e incorporados al sistema.

Una vez completa la implementación se procesó el corpus de 65528 textos utilizando el corrector y se calculó la cantidad de errores de cada tipo en cada texto. Estas cantidades fueron utilizadas como *features* junto con el largo del texto para entrenar un clasificador. Se replicó la metodología de un reporte que utilizó el mismo corpus para crear un clasificador que utiliza *Bag of Words*. El clasificador entrenado consigue resultados un 5 % inferiores en promedio al clasificador del reporte, pero no posee la dependencia al ejercicio en particular que se tiene al utilizar BoW.

Dado que los puntajes en el corpus no se encuentran balanceados, se optó por agrupar los puntajes en tres categorías (baja, media y alta). El modelo entrenado es bueno identificando la diferencia entre un texto bajo y uno alto y si bien un 85 % de los textos en la fraja media son clasificados de forma correcta, las distinciones entre las franjas baja-media y media-alta no resultan ser tan claras.

Es importante también tener en cuenta que si bien el modelo entrenado no se encuentra tan "atado" al ejercicio particular como uno que utilice BoW, no es completamente independiente. Nuestro modelo necesita ejercicios que aunque pueden ser sobre tópicos completamente diferentes, lleven al estudiante a construir estructuras gramaticales similares a las encontradas en el corpus. Debemos recordar que el corrector corrige oraciones declarativas y si el ejercicio le indicara al estudiante

que escriba oraciones interrogativas por ejemplo, ni el corrector, ni el modelo lograrán buenos resultados.

6.3. Trabajo a futuro

Se listan a continuación posibles mejoras o líneas de trabajo para potenciar la usabilidad de la herramienta desarrollada.

- Abarcar más tipos de error detectados/corregidos en los textos. Sería valioso incluir en la detección/corrección de errores, el hecho de señalar cuando el sujeto o el verbo de la oración están faltando (*Missing subject* y *Missing verb*).
- Un tipo de error que ocurre frecuentemente es uno al que llamamos *Pronoun reference*. Este error ocurre cuando los estudiantes utilizan el pronombre "he" en lugar del pronombre "she". Sabemos en este ejercicio en particular, que los estudiantes deben describir a Andrea, por lo que deberían utilizar el pronombre personal femenino. Este tipo de error se puede considerar como un error semántico, ya que la estructura gramatical de la oración es correcta sin importar cuál pronombre personal utilicen. Se podría investigar el uso de un conjunto de respuestas correctas al ejercicio para extraer información relevante e identificar de qué manera se comportan las nuevas respuestas en comparación.
- Contar con más ejemplos corregidos a mano para realizar una mejor evaluación de la herramienta.
- La implementación actual del corrector permite invocarlo desde línea de comandos, generando archivos con el marcado y la corrección. El corrector puede ser utilizado como el motor de una aplicación que posea una interfaz amigable para que el maestro pueda corregir textos. Otra posible mejora está relacionada con la usabilidad de la herramienta. Al crear una interfaz donde la salida del texto corregido incluya etiquetas con colores, el incentivo de los maestros y estudiantes por utilizar la herramienta será mayor.
- Incorporar heurísticas para detectar errores en oraciones interrogativas e imperativas.
- Las heurísticas desarrolladas pueden ser fácilmente modificadas para introducir errores (de los tipos de errores estudiados) en textos correctos. Esto puede ser útil para aumentar la cantidad de determinados tipos de error en un corpus sin depender de encontrar textos que los posean de forma natural.
- Estudiar la posibilidad de construir un corpus *silver standard* utilizando las heurísticas para entrenar modelos de aprendizaje automático secuencia a secuencia (*seq2seq*). Ya sea como un problema de traducción automática o buscar un enfoque similar al utilizado en GECToR [28].

Glosario

BERT Técnica del Procesamiento del Lenguaje Natural que utiliza el contexto de una palabra en un texto para determinar información acerca de esta. 27–29, 32, 45, 46, 51, 52, 56, 81

cross-validation Técnica utilizada al entrenar modelos de aprendizaje automático utilizando distintos subconjuntos de validación para garantizar una mayor robustez del modelo. 30

dataset Conjunto de datos. 30, 31

etiquetador morfológico Herramienta que se encarga de asignar de forma automática una etiqueta a cada palabra indicando su clase morfológica. 14, 50, 60, 82

feature Propiedad o característica medible utilizada para entrenar modelos de aprendizaje automático. 19, 21, 29, 70, 71, 74–79, 82

mal-rules Técnica que busca crear reglas gramaticales con el fin de identificar estructuras sintácticas que posean errores específicos. 51, 53, 56

n-gramas Secuencias de n palabras consecutivas en un texto. Llamamos bigramas y trigramas a los 2-gramas y 3-gramas respectivamente. 21, 24–27, 43, 72

open-source Categoría de software donde los usuarios poseen los derechos de utilizar, estudiar, cambiar, y distribuir el software o el código fuente sin restricción alguna. 15, 43

overfitting Fenómeno que puede ocurrir al entrenar modelos de aprendizaje automático donde el modelo logra identificar de buena manera los ejemplos antes vistos, pero falla al intentar identificar nuevos elementos. No se logra generalizar en la solución del problema. 30, 77

oversampling Técnica que se utiliza al tener conjuntos de datos desbalanceados. En este caso, se busca aumentar la cantidad de elementos de las clases más pequeñas creando copias de sus elementos. 78

parser Herramienta que se encarga de determinar la estructura sintáctica de una oración. 14, 16–18, 20, 49–51, 54, 56, 82

pipeline Estrategia de diseño para manipular información. Se compone de distintos módulos donde la entrada de un módulo es la salida del anterior y cada módulo se encarga de realizar una tarea específica. 13, 14, 42, 47, 54, 82

Reddit Sitio web basado en foros, donde los usuarios discuten las noticias y tópicos de interés. 29

silver standard Conjunto de datos anotados, en general por herramientas automáticas donde se asume un cierto nivel de ruido en los datos. 83

spacy Librería con herramientas para realizar tareas pertenecientes al Procesamiento del Lenguage Natural. 14, 15, 47, 48, 54, 64, 68, 82

spellchecker Corrector ortográfico. 44–47

tagset Conjunto de etiquetas, existen distintos estándares y por lo tanto distintos conjuntos. Diferentes conjuntos ofrecen distintos niveles de granularidad. 15

token Unidad generada en el proceso de tokenización, generalmente representa palabras, números, signos de puntuación, etc. 14, 29, 39–41, 54, 55, 60, 61, 97

tokenizador Herramienta que se encarga de dividir automáticamente un texto en tokens, donde un token generalmente es una palabra. 14

treebank Corpus de texto anotado con las estructuras sintácticas o semánticas de las oraciones. 15–17

undersampling Técnica que se utiliza al tener conjuntos de datos desbalanceados. En este caso, se busca disminuir la cantidad de elementos de las clases más grandes eliminando algunos de sus elementos. 78

Lista de acrónimos

ANEP Administración Nacional de Educación Pública. 1, 9, 10

ASR Automatic Speech Recognition. 13

BoW Bag of Words. 31, 71, 72, 74, 76–79, 82

CALI Computer Assisted Language Instruction. 20

CALL Computer Assisted Language Learning. 20

CEFR Common European Framework of Reference for Languages. 45, 46, 51, 80

CEI Ceibal En Inglés. 9

CoNLL Conference on Computational Language Learning. 22

ESL English as a Second Language. 56

GEC Grammatical Error Correction. 19–22, 81

IA Inteligencia Artificial. 12, 29

ME Máxima Entropía. 71, 72, 75

MLP Multilayer Perceptron. 71, 76, 77

MT Machine Translation. 23

NB Naive Bayes. 71

NER Named Entity Recognition. 14, 18, 19, 54, 82

NLTK Natural Language Toolkit. 41

NMT Neural Machine Translation. 21

PLN Procesamiento del Lenguaje Natural. 1, 10–15, 20–22, 28, 56, 71, 81, 82

POS Part Of Speech. 14, 17–19, 21, 50, 54, 55, 64, 68

PPL Programa de Políticas Lingüísticas. 1, 10

RF Random Forest. 77

SMT Statistical Machine Translation. 21

SVM Support Vector Machines. 71

TTS Text-to-Speech. 13

UdelaR Universidad de la República. 1

XML Extensible Markup Language. 39, 41

Bibliografía

[1] S. Ailani, A. Dalvi, and I. Siddavatam. Grammatical error correction (gec): Research approaches till now. *International Journal of Computer Applications*, 178:1–3, 08 2019.

[2] E. Brill. A simple rule-based part of speech tagger. In *Third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy, Mar. 1992. Association for Computational Linguistics.

[3] B. K. Britton. Lexical ambiguity of words used in english text. *Behavior Research Methods & Instrumentation*, 10(1):1–7, Jan 1978.

[4] C. Bryant, M. Felice, Ø. E. Andersen, and T. Briscoe. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[5] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models, 2020.

[6] M. Catt and G. Hirst. An intelligent cali system for grammatical error diagnosis. *Computer Assisted Language Learning*, 3(1):3–26, 1990.

[7] L. M. da Costa, F. Bond, and X. He. Syntactic well-formedness diagnosis and error-based coaching in computer assisted language learning using machine translation. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 107–116, 2016.

[8] A. DeCapua. *Grammar for teachers*. Springer, 2010.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.

[11] R. Garside. The CLAWS word-tagging system. In R. Garside, G. Leech, and G. Sampson, editors, *The Computational Analysis of English: a corpus-based approach*, pages 30–41. Longman, London, 1987.

[12] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models, 2020.

[13] Y. Goldberg. Assessing bert's syntactic abilities, 2019.

[14] Y. Goldberg and G. Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[15] R. Grundkiewicz and M. Junczys-Dowmunt. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 284–290, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[16] G. Herrera. Reporte de pruebas de pln para pruebas de inglés. Technical report, InCo, UdelaR, Montevideo, Uruguay, 2019.

[17] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, Apr. 1998.

[18] A. S. Hornby. *Oxford Advanced Learner's Dictionary of Current English*. Oxford University Press, ninth edition, 2014.

[19] D. Jurafsky and H. Martin. Speech and language processing (draft). *URL: https://web.stanford.edu/~jurafsky/slp3*, 2020.

[20] D. Jurafsky and J. H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009.

[21] K. Kukich. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439, Dec. 1992.

[22] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[23] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[24] R. Nagata, E. Whittaker, and V. Sheinman. Creating a manually error-tagged and shallow-parsed learner corpus. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1210–1219, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[25] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[26] H. T. Ng, S. M. Wu, Y. Wu, C. Hadiwinoto, and J. Tetreault. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics.

[27] D. Nicholls. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581, 2003.

[28] K. Omelianchuk, V. Atrasevych, A. Chernodub, and O. Skurzhanskyi. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online, July 2020. Association for Computational Linguistics.

[29] Oxford University Press. Oxford 5000 wordlist, aug 2020. https://www.oxfordlearnersdictionaries.com/us/wordlists/oxford3000-5000.

[30] D. Schneider and K. F. McCoy. Recognizing syntactic errors in the writing of second language learners. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1198–1204, Montreal, Quebec, Canada, Aug. 1998. Association for Computational Linguistics.

[31] F. Stahlberg and S. Kumar. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online, Apr. 2021. Association for Computational Linguistics.

[32] M. Swan and C. Walter. *Oxford English grammar course*. Oxford University Press, 11 edition, 2011.

[33] A. Szanser. Automatic error-correction in natural languages. *Information Storage and Retrieval*, 5(4):169–174, 1970.

[34] W. L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[36] Y. Wang, Y. Wang, J. Liu, and Z. Liu. A comprehensive survey of grammar error correction. *arXiv preprint arXiv:2005.06600*, 2020.

[37] Y. Wang and H. Zhao. A light rule-based approach to English subject-verb agreement errors on the third person singular forms. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters*, pages 345–353, Shanghai, China, Oct. 2015.

[38] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.

Anexo A

Rúbrica corrección de producción de la prueba adaptativa

Tabla A.1: Criterios de corrección

Descripción	**Nota**	**Nivel**
No hay respuesta / el alumno no entendió la consigna / el alumno responde completamente en español.	0	A0
Producción mínima. Escribe palabras (o una sola) sueltas en inglés, generalmente con errores de ortografía, pero las mismas no están relacionadas con la tarea propuesta o no forman una idea clara. El alumno recurre a la lengua materna (L1) para formular oraciones, en caso de hacerlo.	0	A0
Intenta comunicarse, pero básicamente escribe palabras o frases sueltas en inglés, las cuales suelen tener errores de ortografía. Dichos errores de ortografía y sintaxis son frecuentes, lo cual interfiere en la comprensión del texto. No llega a formular una oración coherente y suele recurrir a L1 para poder completar ideas (suele recurrir a interlengua). Suele haber un uso inadecuado del sistema de puntuación.	1	A0
En general logra comunicarse escribiendo palabras. Si intenta incluir una (o más) oraciones, las mismas suelen tener errores de ortografía y sintaxis que interfieren en la comprensión del texto. El alumno, por momentos, puede necesitar usar L1 para terminar de formular ideas. Puede no haber un uso apropiado del sistema de puntuación.	2	A1-
Continúa en la próxima página...		

Descripción	Nota	Nivel
Completa y/o escribe oraciones en L2 sin recurrir a su lengua materna. Intenta proporcionar algunos detalles tales como: descripción física de personas, gustos y preferencias, vestimenta, habilidades y lugares. El alumno puede cometer errores de ortografía y sintaxis los cuales pueden llegar a interferir mínimamente con la comprensión del texto. Pueden aparecer elementos de cohesión básicos (and). En general, el alumno usa el sistema de puntuación correctamente, aunque puede haber algunos errores al respecto.	3	A1+
Completa y/o escribe oraciones en inglés sin recurrir a L1 y describe dando detalles de la mayoría de los puntos sugeridos tales como: descripción física de personas, gustos y preferencias, vestimenta, habilidades y lugares. El alumno no comete errores significativos de sintaxis que puedan interferir con la comprensión de texto. Se observa el uso de conjunciones (and, but) entre adjetivos y verbos, pero no se observan elementos de cohesión interoracionales. Hay un uso adecuado (puede haber algún error) del sistema de puntuación.	4	A2-
Completa y/o escribe oraciones en inglés sin recurrir a L1 y describe dando detalles tales como: descripción física de personas, gustos y preferencias, vestimenta, habilidades y lugares. Puede intentar dar detalles más allá de los temas sugeridos, usando vocabulario relevante a la propuesta. Se observan elementos de cohesión tales como conjunciones (and, but, or) y conectores discursivos (because). De haber algún error de sintaxis o de ortografía, no interfiere con la comprensión del texto. Hay un uso correcto del sistema de puntuación.	5	A2+
Completa y/o escribe oraciones en inglés sin recurrir a L1 y describe dando detalles tales como: descripción física de personas, gustos y preferencias, vestimenta, habilidades y lugares. Además incorpora el uso de diferentes tiempos verbales o demuestra manejo de diversas estructuras, no se limita a reproducir las estructuras incluidas en la prueba escrita. Usa vocabulario variado y relevante a la propuesta. Puede extenderse más allá de las ideas sugeridas. Se observan elementos de cohesión tales como conjunciones (and, but, or) y conectores discursivos (because, so). De haber algún error de sintaxis o de ortografía, no interfiere con la comprensión del texto. Hay uno uso correcto del sistema de puntuación.	6	B1

Anexo B

Experimentos utilizando el corrector

Analizaremos el comportamiento del corrector en una combinación de ejemplos extraídos del muestreo de desarrollo, de evaluación y algún ejemplo creado artificialmente para analizar cuáles tipos de errores se detectan, cuáles no son detectados y bajo qué circunstancias encontramos estos casos.

Para cada ejemplo se muestra el texto original, y el texto obtenido como salida del corrector luego de procesar el texto con los *tags* correspondientes. Para facilitar la lectura e interpretación de los resultados, los *tags* se marcan con diferentes colores y la figura B.1 muestra cada *tag* con su color y definición correspondiente. Con el mismo propósito, se omitió el parámetro "corr" en las etiquetas que luego es utilizado por el corrector con el fin de devolver el texto corregido.

<E_S> ⟶ Error de ortografía (*Spelling*).
<E_BOS> ⟶ Omisión de mayúscula al comienzo de la oración (*Beg. Of Sentence*).
<E_PI> ⟶ Omisión de mayúscula en el pronombre "I" (*Pronoun I*).
<E_PN> ⟶ Omisión de mayúscula en nombre propio (*Proper Name*).
<E_SVA> ⟶ Error de concordancia sujeto-verbo (*Subject-Verb Agreement*).
<E_VF> ⟶ Error de forma del verbo (*Verb Form*).
<E_MA> ⟶ Error de omisión de artículo (*Missing Article*).
<E_UA> ⟶ Error de artículo innecesario (*Unnecessary Article*).

Figura B.1: Esquema de *tags* y tipo de error.

Ejemplo extraído del muestreo de entrenamiento - ID: 8401

She is 14 years old.
she is tall thin and she has yellow and short hair. In this moment she is riding her bike with her dog.
Andrea is wering a pinck t-shirt and shorts.
she like the pizza but she don't like the apple, Andrea like reding but she don't like to sing, she like math but she don't like fishing. in the morning she has breakfast, for lunch she has meat and salad

She is 14 years old.
<E_C-BOS> *she* </E_C-BOS> *is tall thin and she has yellow and short hair.*
In this moment she is riding her bike with her dog.
Andrea is <E_S orig="wering" sug="['wearing', 'being', 'during', 'working', 'spring']" > *wearing* </E_S> *a* <E_S orig="pinck" sug="['pink', 'pick', 'link', 'since', 'pack']" > *pink* </E_S> *t-shirt and shorts.*
<E_C-BOS> *she* </E_C-BOS> <E_SVA> *like* </E_SVA> *the pizza but she do not* <E_SVA> *like* </E_SVA> *the apple, Andrea* <E_SVA> *like* </E_SVA> <E_S orig="reding" sug="['reading', 'riding', 'being', 'rating', 'wedding']" > *reading* </E_S> *but she do not* <E_SVA> *like* </E_SVA> *to sing, she* <E_SVA> *like* </E_SVA> *math but she do not* <E_SVA> *like* </E_SVA> *fishing.*
<E_C-BOS> *in* </E_C-BOS> *the morning she has breakfast, for lunch she has meat and salad*

Figura B.2: Ejemplo con ID: 8401

Este ejemplo (figura B.2) fue calificado con un puntaje de cuatro. Se encontraron tres errores de ortografía (**E_S**) los cuales fueron detectados y corregidos satisfactoriamente. Las palabras "wering", "pinck" y "reding" fueron remplazadas por las palabras "wearing", "pink" y "reading" respectivamente.

Las palabras que comienzan oraciones y no comienzan con mayúscula fueron detectadas y marcadas como errores de *Beginning of Sentence* (**E_C-BOS**).

Se encontraron también errores de concordancia sujeto-verbo (**E_SVA**) y entre estos errores encontramos dos tipos distintos de situaciones que los provocan. El primer tipo sucede en "she like ...", aquí el verbo principal es donde se genera la falta de concordancia, ya que este verbo debería estar en la tercera persona del singular. El otro tipo sucede en la estructura "she do not like ...", donde el verbo que no concuerda en este caso es el verbo auxiliar "do", el cual debería ser "does". Si bien el etiquetado se realiza siempre sobre el verbo principal, el componente de **SVA** del corrector tiene la información necesaria para distinguir a cuál de estos dos casos nos estamos enfrentando.

Ejemplo extraído del muestreo de entrenamiento - ID: 10246

She is andrea, she is fourteen years old.
In this moment she has a sun glasses, a pair of trainers, a short and a pink bluse.
She likes reading and eating pizza, but she not like (“cantar”) and eat apple’s.
She likes drawing and ride in your bike with your little dog, she hate fishing

She is <E_C-PN> *andrea* </E_C-PN> *, she is fourteen years old.*
In this moment she has a sun glasses, a pair of trainers, a short and a pink <E_S orig=“bluse” sug=“[‘blue’, ‘blouse’, ‘use’, ‘house’, ‘close’]” > *blue* </E_S> *.*
She likes reading and eating pizza, but she not <E_SVA> *like* </E_SVA> *(“* <E_S orig=“cantar” sug=“[‘center’, ‘cancer’]” > *center* </E_S> *”) and* <E_VF> *eat* </E_VF> *apple’s.*
She likes drawing and <E_VF> *ride* </E_VF> *in your bike with your little dog, she* <E_SVA> *hate* </E_SVA> *fishing*

Figura B.3: Ejemplo con ID: 10246

Este ejemplo (figura B.3) también fue calificado con un puntaje de cuatro. Podemos ver un error de nombre propio sin mayúscula (**E_C-PN**) marcado en la palabra “andrea”. Se encuentran dos errores de ortografía (**E_S**), uno de ellos es la palabra en español “cantar”, la cual se sustituye por la palabra en inglés “center”. El otro error de ortografía es en “bluse”, la cual se corrige erróneamente a “blue”, cuando la segunda palabra en la lista de sugerencias (“blouse”) es la corrección adecuada.

Se detectan dos errores de forma del verbo (**E_VF**). En el primer caso se debería utilizar el infinitivo pero se omite la palabra “to”. En el segundo caso la forma adecuada del verbo es un gerundio (“riding”) por la regla de paralelismo, ya que previamente se utilizó esta forma del verbo en la oración.

El error de concordancia (**E_SVA**) en “but she not like ...” es interesante porque la falta del verbo auxiliar “does” en la construcción dispara la detección entre el sujeto “she” y el verbo principal “like”. En este caso, el error se encuentra en la falta del verbo auxiliar y si corrigiéramos esta oración según como fue marcada, obtendríamos “but she not likes ...” por lo que no se corregiría el error de forma adecuada.

En el ejemplo existe otro tipo de error en “apple’s”, el cual suponemos es un error al intentar escribir el plural de este sustantivo. Este tipo de errores con el uso del apóstrofo no sucede frecuentemente en el muestreo de entrenamiento por lo que no fue considerado en el universo de errores que se buscan detectar.

Ejemplo extraído del muestreo de entrenamiento - ID: 60359

I like a book i dont like music i like a pizza i dont like apple i like study i dont like fishing

I like a book <E_C-PI> *i* </E_C-PI> *do* <E_S orig="nt" sug="['not']" > *not* </E_S> *like music* <E_C-PI> *i* </E_C-PI> *like a pizza* <E_C-PI> *i* </E_C-PI> *do* <E_S orig="nt" sug="['not']" > *not* </E_S> *like* <E_MA> *apple* </E_MA> <E_C-PI> *i* </E_C-PI> *like* <E_VF> *study* </E_VF> <E_C-PI> *i* </E_C-PI> *do* <E_S orig="nt" sug="['not']" > *not* </E_S> *like fishing*

Figura B.4: Ejemplo con ID: 60359

Este ejemplo (figura B.4) fue calificado con un puntaje de dos. Podemos ver que al compararlo con los ejemplos anteriores (calificados como cuatros), este es menos extenso y se encuentra compuesto técnicamente por una sola oración (no posee puntuación alguna).

También presenta una característica interesante desde el punto de vista semántico. Este ejemplo se encuentra escrito en la primera persona, pero sabemos que el objetivo del ejercicio propuesto a los estudiantes es describir los gustos y actividades de Andrea, por lo que se espera que los estudiantes escriban en la tercera persona del singular. Es difícil distinguir si el error sucedió producto de una confusión al entender el problema, o si se debe a falta de conocimiento sobre el uso de la primera/tercera persona en inglés. De igual manera, el corrector no tiene la capacidad de detectar este tipo de situaciones más semánticas en naturaleza.

En cuanto a los errores marcados, en este ejemplo tenemos un caso de texto donde el pronombre "I", no se encuentra en mayúscula, estos errores se marcan utilizando la etiqueta **E_C-PI**. Los errores de ortografía (**E_S**) corresponden todos a la misma palabra "dont". En estos casos, al analizar el texto, el segmentador al particionar en *tokens* produce las *tokens* "do" y "nt", y esta última es corregida a "not".

Luego podemos observar el marcado de un error de tipo **E_MA** en la palabra "apple" donde el corrector indica que debería ser "an apple" ya que se trata de un sustantivo singular contable. En realidad, estudiándolo semánticamente la corrección debería ser el sustantivo en plural "apples" ya que se está hablando del gusto por la fruta, pero este detalle no es detectado por el corrector.

Por último, tenemos un error de forma verbal (**E_VF**) al intentar utilizar el infinitivo "to study" pero olvidar "to".

Ejemplo extraído del muestreo de validación - ID: 14612

She is Andrea.She is 14 years old.She has got a dog.She like study but she doesn't like sing.He likes pizza but she doesn't like apple.
In the morning she gets up , she has breakfast , she goes to English Class at nine o'clock. In the afternoon she goes home , she goes to the school , she works , she plays during the break.
In the evening she goes to bed because is tired.

She is Andrea.
She is 14 years old.
She has got a dog.
She <E_SVA> *like* </E_SVA> <E_VF> *study* </E_VF> *but she does not like* <E_VF> *sing* </E_VF> .
He likes pizza but she does not like <E_MA> *apple* </E_MA> .
In the morning she gets up, she has breakfast, she goes to English Class at nine o'clock. In the afternoon she goes home, she goes to the school, she works, she plays during the break. In the evening she goes to bed because is tired.

Figura B.5: Ejemplo con ID: 14612

Este ejemplo (figura B.5), calificado con un puntaje de cuatro no posee una gran cantidad de errores. Los errores detectados (**E_SVA**, **E_VF**, **E_MA**) son ocurrencias comunes de estos errores en los textos del corpus. El error de concordancia en particular **E_SVA**, parece deberse a un descuido ya que es la única instancia en el texto donde se ejecuta de manera incorrecta sobre un total de 15 instancias de concordancia sujeto-verbo. No deja de ser sin embargo un error y es detectado de forma correcta.

Podemos encontrar en este ejemplo una confusión en el uso del pronombre en la oración "He likes pizza but ...". Nuevamente, al conocer la naturaleza del ejercicio, y sumado a que el resto de los pronombres utilizados en este texto son "she" podemos deducir que existe aquí un error. El corrector no cuenta con información acerca del ejercicio que se intenta resolver por lo que no es posible detectar este tipo de errores.

Otro error que podemos encontrar en este ejemplo sucede en la última oración, "... she goes to bed because is tired.". Aquí falta el sujeto "she", este tipo de error no se encontró entre los más frecuentes a la hora de determinar el dominio de errores a atacar, por lo que no fue considerado como uno de los candidatos.

Ejemplo extraído del muestreo de entrenamiento - ID: 57745

Like a books i don't like music
Like pizza i don't like apple
Like school i don't like fish

Like a <E_UA> *books* </E_UA> <E_C-PI> *i* </E_C-PI> *do not like music*
Like pizza <E_C-PI> *i* </E_C-PI> *do not like* <E_MA> *apple* </E_MA>
Like school <E_C-PI> *i* </E_C-PI> *do not like fish*

Figura B.6: Ejemplo con ID: 57745

Este ejemplo (figura B.6) fue calificado con un puntaje de 1. Nuevamente podemos observar cómo baja la complejidad de las construcciones gramaticales en los textos con menor puntaje.

Este texto, al igual que el anterior se encuentra en su mayor parte en primera persona, y al utilizar el pronombre "I", se olvida que el mismo debe utilizarse en mayúscula en inglés (un error común, dado que no tenemos una regla de este estilo en el español).

El texto está compuesto por tres oraciones (estamos considerando tres oraciones, porque no hay puntuación para generar oraciones) y todas ellas comienzan con "like". Esto es correcto desde el punto de vista gramatical, pero probablemente no es lo que el/la estudiante quiso expresar. Es más probable que se haya olvidado de utilizar el sujeto.

Por último, se detecta una instancia de un error **E_UA**, dado que en "Like a books", al ser el sustantivo plural no necesita el artículo "a".

Ejemplo extraído del muestreo de entrenamiento - ID: 10248

She is Andrea, she is 14 years old, she is thin and her hair is blond .
He likes to read, to write, he does not like the apple, neither sing nor fish, his food preferred is pizza.
Finally, the sports routine he practices in riding a bicycle, and he has a little dog.

She is Andrea, she is 14 years old, she is thin and her hair is blond.
He likes to read, to write, he does not like the apple, neither <E_VF> *sing* </E_VF> *nor* <E_VF> *fish* </E_VF> *, his food preferred is pizza.*
Finally, the sports <E_MA> *routine* </E_MA> *he practices in riding a bicycle, and he has a little dog.*

Figura B.7: Ejemplo con ID: 10248

Este ejemplo (figura B.6) fue calificado con un puntaje de 5. Nuevamente encontramos la confusión en el uso del pronombre masculino cuando el ejercicio planteado trata de una persona de género femenino. Los errores de forma del verbo son correctamente detectados.

Por último, en la oración final del texto el corrector automático detecta incorrectamente un error sobre el uso de artículos. Esto se debe a que la palabra "sports" no es detectada por el *tagger* como adjetivo sino que es etiquetada como sustantivo plural, provocando que se detecte incorrectamente la falta de artículo en el sustantivo singular "routine".

Ejemplo artificial

i like to drink orange juice but shi lke apple juice better.
I always eat in the afternoon. I like eat in the afternoon.

<E_C-BOS> *i* </E_C-BOS> *like to drink* <E_MA> *orange* </E_MA>
juice but <E_S orig="shi" sug="['she']" > *she* </E_S> <E_SVA>
<E_S orig="lke" sug="['like', 'lake', 'lie', 'the', 'be']" > *like* </E_S> </E_SVA>
apple juice better.
I always eat in the afternoon.
I like <E_VF> *eat* </E_VF> *in the afternoon.*

Figura B.8: Ejemplo artificial

Este ejemplo (figura B.8) no posee una calificación al ser creado artificialmente para mostrar otros comportamientos del corrector.

Hay una situación particular, que sucede cuando "i" ocurre como la primera palabra de una oración. Este error se encuentra contemplado por dos tipos de errores distintos en el corrector, **E_C-BOS** y **E_C-PI**, los errores que verifican la mayúscula al comienzo de la oración y sobre el pronombre "I" respectivamente. Se tomo la decisión de marcar este tipo de error como **E_C-BOS** ya que resulta más sencillo de entender en el contexto.

Como en el ejemplo anterior el *tagger* etiqueta incorrectamente "orange" como un sustantivo singular cuando en realidad está cumpliendo la función de "adjetivo" del sustantivo no contable "juice". Esto lleva al corrector a buscar un artículo anterior al sustantivo pero al no encontrarlo se marca el error de falta de artículo.

Luego tenemos un caso donde ambos el sujeto y el verbo tienen errores de ortografía (**E_S**), ambos son detectados correctamente y esto lleva a que se pueda detectar correctamente un "nuevo" error, esta vez de concordancia (**E_SVA**) entre estos elementos. Detectar este error de concordancia no sería posible si los errores de ortografía no fueran corregidos en primer lugar.

Por último, tenemos un ejemplo de dos oraciones similares donde una posee un error de forma verbal (**E_VF**) ya que el uso del infinitivo luego de un verbo como "like" requiere el uso de "to" por lo que la construcción correcta sería "... like to eat ...". Mientras que en la primera oración, al existir el adverbio "always", es correcto utilizar el verbo "eat" de la forma "... always eat ...".

Anexo C

Ejecutar el corrector

El corrector fue implementado utilizando el lenguaje de programación *python3*[1] y el código del proyecto se encuentra en el *gitlab* de la Facultad de Ingeniería en la URL: `https://gitlab.fing.edu.uy/romina.brown/pgensinglesruralcorreccion`

Para clonar el repositorio del proyecto utilizamos el comando:

```
$git clone
 ↪  https://gitlab.fing.edu.uy/romina.brown/pgensinglesruralcorreccion.git
```

C.1. Instalar dependencias

Recomendamos antes de continuar, instalar un ambiente virtual (*virtual environment*) para aislar las dependencias del resto de los paquetes de *python* que puedan existir en el sistema.

Para instalar un ambiente virtual, se puede utilizar la librería *anaconda*[2] o la librería *venv*[3]. Ambas proveen esta funcionalidad.

Una vez creado el ambiente virtual podemos instalar las dependencias necesarias evitando posibles conflictos con dependencias existentes.

```
$pip install -m requirements.txt
```

El archivo "requirements.txt" se encuentra en el repositorio y posee una lista de todas las dependendencias necesarias, de igual manera estas dependencias se encuentran listadas en la sección C.3.

Una vez instaladas las dependencias, el último paso es descargar el modelo utilizado por *spacy*. Esto lo realizamos de la siguiente manera:

[1] `https://www.python.org/doc/`
[2] `https://www.anaconda.com/products/individual`
[3] `https://docs.python.org/3/tutorial/venv.html`

```
$python -m spacy download en_core_web_sm
```

Para verificar que el corrector ejecute, se recomienda correr el *script* "corrector/test/test_correccion.py", lo cual se puede lograr navegando hacia el directorio "corrector/test/" y haciendo:

```
$python test_correccion.py
```

Este *script* simplemente ejecuta la corrección de un texto predeterminado e imprime el marcado y la corrección en pantalla.

C.2. Marcado de textos

Para procesar un conjunto de textos, existe el directorio "corrector/entradas/" en el cual se deben agregar los archivos correspondientes a los textos (un archivo por texto que se quiera procesar) donde no importa el nombre de los archivos siempre y cuando tengan la extensión ".txt".

El *script* "correccion_textos.py" se encarga de procesar los textos, informa en pantalla cuáles textos fueron procesados de forma correcta y crea en el directorio "corrector/salidas/" los archivos correspondientes al marcado y corrección de los textos procesados.

Para ejecutar el *script* "corrector/correccion_textos.py" se navega hasta el directorio "corrector/" y ejecutamos:

```
$python correccion_textos.py
```

C.3. Lista de dependencias

Las dependencias que existen en el archivo "requirements.txt" son:

```
attrs==21.2.0
blis==0.7.4
brotlipy==0.7.0
CacheMan==2.1.0
catalogue==1.0.0
certifi==2021.5.30
cffi==1.14.6
charset-normalizer==2.0.4
click==8.0.3
cryptography==35.0.0
cymem==2.0.5
future==0.18.2
idna==3.2
importlib-metadata==4.8.1
joblib==1.0.1
jsonschema==3.2.0
lemminflect==0.2.2
murmurhash==1.0.5
nltk==3.5
numpy==1.19.5
pip==21.2.2
plac==0.9.6
preshed==3.0.5
psutil==5.8.0
pycparser==2.20
pyOpenSSL==21.0.0
pyrsistent==0.17.3
PySocks==1.7.1
regex==2021.8.3
requests==2.26.0
setuptools==58.0.4
six==1.16.0
spacy==2.3.4
srsly==1.0.5
symspellpy==6.5.2
thinc==7.4.5
tqdm==4.62.3
typing-extensions==3.10.0.2
Unidecode==1.3.2
urllib3==1.26.7
wasabi==0.8.2
wheel==0.37.0
zipp==3.6.0
```