Normalización de Texto en Español de Argentina

Facultad de Matemática, Astronomía, Física y Computación

Universidad Nacional de Córdoba

Alan Bracco

Director: Franco M. Luque

Resumen

En la actualidad la cantidad de datos que consume y genera una sola persona es gigantesca. Los datos cada vez son más, ya que cualquiera puede generarlos. Esto trae consigo un aumento en el ruido que hay en esos datos. Es por eso que el texto de las redes sociales se caracteriza por ser ruidoso, lo que es un problema cuando se quiere trabajar sobre ellos.

En este trabajo construimos un corpus de tweets en español de Argentina. Recolectamos un conjunto grande de tweets y luego los seleccionamos manualmente para obtener una muestra representativa de los errores típicos de normalización. Luego, definimos criterios claros y explícitos de corrección y los utilizamos para proceder a la anotación manual del corpus.

Además, presentamos un sistema de normalización de texto que trabaja sobre tweets. Dado un conjunto de tweets como entrada, el sistema detecta y corrige las palabras que deben ser estandarizadas. Para ello, utiliza una serie de componentes como recursos léxicos, sistemas de reglas y modelos de lenguaje.

Finalmente, realizamos experimentos con diferentes corpus, entre ellos el nuestro, y diferentes configuraciones del sistema para entender las ventajas y desventajas de cada uno.

Palabras clave: Procesamiento de lenguaje natural, normalización de texto, Twitter, recursos lingüísticos, español de Argentina, texto ruidoso, redes sociales.

Abstract

Nowadays, the amount of data consumed and generated by only one person is enormous. Data amount keeps growing because anyone can generate it. This brings along an increment of noisy data. That is why social network text is noisy, which is a problem when it is needed to work on it.

Here, we built a corpus of tweets in argentinian spanish. We collected a big set of tweets and we selected them manually to obtain a representative sample of common normalization errors. Then, we defined explicit and clear correction criteria and we used it to continue with the manual corpus annotation.

Besides, we present a text normalization system that works on tweets. Given a set of tweets as input, the system detects and corrects words that need to be standardized. To do that, it uses a group of components as lexical resources, rule-based systems and language models.

Finally, we made some experiments with different corpus, among them, the one we built, and different system configurations to understand each one's advantages and disadvantages.

Key words: Natural language processing, text normalization, Twitter, linguistic resources, argentine spanish, noisy text, social networks.

Agradecimientos

Agradezco infinitamente a mis padres por haberme acompañado y apoyado desinteresadamente desde siempre y sobre todo durante todos estos años de estudio.

A mi director Franco Luque, con quien pude compartir diferentes momentos en la carrera, y también ahora que la estoy finalizando, que sin su guía esto no hubiera sido posible.

Finalmente agradecer a todos los compañeros y amigos que me han ayudado de alguna manera durante el cursado y han hecho que la carrera sea mucho más amigable y llevadera.

Índice

1. Introducción

En los últimos años, la cantidad de datos en forma de texto que se puede encontrar libremente en internet ha crecido enormemente. Casi cualquier persona puede aportar directa o indirectamente a esta cantidad de datos que se generan. El medio típico para la generación de contenido es el de las redes sociales.

Esta gran, y aún creciente, cantidad de datos es una fuente útil para la realización de investigaciones y aplicaciones en Procesamiento de Lenguaje Natural (PLN).

En las redes sociales, los contenidos producidos tienen la particularidad de ser ruidosos. Esto es, que contienen diferentes deformaciones del lenguaje como informalidad, errores ortográficos frecuentes, uso de abreviaciones, acrónimos, emoticones, énfasis en palabras, expresiones propias de las redes, errores de tipeo, etc. Estas características son una desventaja, ya que agregan dificultades a la hora de procesar los textos automáticamente.

Una posible solución al problema de los datos ruidosos es la normalización de texto. Su objetivo es mejorar la calidad de los datos, transformándolos a una forma estandarizada del lenguaje. Luego, el texto normalizado puede ser utilizado para la realización de otras tareas como análisis de sentimientos, traducción automática, resúmenes automáticos, etc.

La normalización de textos de redes sociales y, en particular, de tweets, ha sido extensivamente estudiada para el idioma inglés, como en (Han & Baldwin 2011). Para el idioma español, los principales trabajos fueron realizados en base a conjuntos de datos de España, publicados en el marco de la competencia Tweet-Norm 2013 (Alegria et al. 2013) organizada por

la SEPLN (Sociedad Española para el Procesamiento de Lenguaje Natural). Estos datos, sin embargo, no contienen ni contemplan los problemas de normalización que se presentan en textos producidos por usuarios de otros países de habla hispana como Argentina.

En este trabajo, estudiamos el problema de normalización de texto prestando especial atención al dominio del lenguaje español utilizado en redes sociales por usuarios de Argentina.

Para ello, creamos un corpus de tweets de Argentina anotado con información de normalización. Este corpus está compuesto de 1200 tweets recolectados en diferentes franjas horarias, y puede servir para entrenar y evaluar sistemas de normalización. Seleccionamos manualmente un subconjunto de tweets que contienen errores típicos de normalización, dejando al menos un tweet de cada tipo de error. Realizamos una anotación manual de las correcciones para las palabras incorrectas, siguiendo una serie de criterios de corrección claros y explícitos definidos por nosotros.

Además, desarrollamos un normalizador de texto. Las principales características del sistema desarrollado provienen del sistema de normalización de (Gamallo et al. 2013), presentado en Tweet-Norm 2013 (Alegria et al. 2013). Utiliza recursos léxicos para clasificar las palabras en correctas e incorrectas, y un sistema basado en reglas para generar palabras normalizadas. También se consulta un modelo de lenguaje para poder decidir cuál es la mejor opción para una palabra dada. De esta forma, el normalizador corregirá de manera automática las palabras incorrectas.

Utilizamos nuestro corpus para realizar diferentes experimentos sobre el normalizador, analizando su desempeño en dos etapas, la de detección y la de corrección. Se utilizan diferentes métricas que nos dan una noción del

comportamiento del sistema en circunstancias variadas. Mediante el análisis de errores, obtenemos nuestras propias conclusiones y proponemos las mejoras que se pueden aplicar sobre el sistema para obtener mejores resultados.

El resto del trabajo se encuentra estructurado de la siguiente manera. En el Capítulo 2 definimos los conocimientos básicos para el entendimiento del problema y solución planteados. En el Capítulo 3 se explica lo relacionado al proceso de construcción de un corpus con tweets en español de Argentina. En el Capítulo 4 se detalla y presenta el sistema de normalización propuesto. En el Capítulo 5 se muestran los experimentos realizados junto con los resultados obtenidos. Finalmente, en el Capítulo 6, concluimos los resultados del trabajo y proponemos formas de seguir trabajando sobre el sistema desarrollado.

2. Preliminares

En este capítulo introducimos los conceptos básicos relacionados al trabajo y que contribuyen al entendimiento del mismo. Comenzaremos definiendo los aspectos más generales y luego nos iremos sumergiendo en aspectos un poco más específicos.

2.1. Procesamiento de Lenguaje Natural

La lingüística es la disciplina que se dedica a estudiar diferentes aspectos del lenguaje como origen, evolución y estructura. Por otra parte, dentro de los campos de la ciencia de la computación, se encuentra el procesamiento de lenguaje natural (PLN o NLP en inglés), que combina la lingüística aplicada con tecnologías computacionales (aprendizaje automático, inteligencia artificial, etc) para hacer posible la comprensión y el análisis de lenguaje humano.

El PLN se compone de diversas tareas que trabajan con el lenguaje en distintos niveles de abstracción. Algunas de estas tareas son el etiquetado gramatical (Part-Of-Speech Tagging en inglés), proceso en el cual se le asigna a cada palabra de un texto su categoría gramatical; análisis de sentimiento, en donde se extrae información subjetiva de los textos; análisis sintáctico (Parsing en inglés), para determinar la estructura sintáctica o árbol sintáctico asociados a una oración; y reconocimiento de entidades nombradas (Named entity recognition en inglés) para localizar y clasificar las entidades nombradas de un texto en categorías predefinidas como personas, organizaciones, lugares, etc.

Cada una de las tareas, puede utilizar diferentes subtareas como la segmentación de texto. Incluye procesos como segmentación de oraciones en donde se divide la secuencia de caracteres que compone el texto en las diferentes oraciones que lo forman. Para ello se deben identificar y desambiguar marcadores de fin de oración (como el signo ".").

Luego para cada una de las oraciones, se realiza la tarea de tokenización, que consiste en dividirlas en secuencias de palabras, también llamadas tokens. A veces puede tomarse como un solo token más de una palabra, por ejemplo un nombre propio con más de una palabra.

A las palabras, se le pueden realizar diferentes procesos. Un ejemplo de estos procesos, es la lematización, en donde dada una palabra, se obtiene su lema, que es la forma canónica o de diccionario de la palabra (por ejemplo, el lema de "comía" es "comer" y el lema de "comiste" también es "comer"). Otro ejemplo es el stemming, que es un método para obtener la raíz (o stem) de una palabra (por ejemplo el stem de "corriste", "correr" y "corrió" es "corr").

2.2. Aprendizaje Automático

El aprendizaje automático (en inglés, machine learning) es la rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas para que las computadoras puedan aprender a generalizar comportamientos a partir de datos suministrados. Extrayendo conocimiento de esta información, podrán decidir sobre cuál es el resultado, y ser capaces de predecir comportamiento futuro a partir de lo que ha ocurrido en el pasado.

Para conseguir esto existen diferentes metodologías de aprendizaje. Las más comunes y conocidas son las de aprendizaje supervisado y aprendizaje no supervisado.

2.2.1. Aprendizaje supervisado

El aprendizaje supervisado es una técnica de aprendizaje automático cuyo objetivo es encontrar una función a partir de datos de entrenamiento. Los datos de entrenamiento son un conjunto de pares que están formados por los valores de entrada y los resultados esperados para esos valores. Tomando como ejemplo los datos de entrenamiento, se genera una función que sea capaz de predecir los resultados ante cualquier elemento válido de entrada.

Dependiendo de si es un problema de regresión o de clasificación, la función retornará un valor numérico o una etiqueta de clase ("label" en inglés) respectivamente.

Existen numerosos algoritmos de aprendizaje supervisado, algunos de ellos ampliamente utilizados. Podemos encontrar las máquinas de vectores de soporte (en inglés Support Vector Machines, SVMs), regresiones lineales, árboles de decisión (en inglés decision trees), bosques aleatorios (en inglés random forest), y redes neuronales entre otros.

En la Figura 1 podemos observar un ejemplo de una función utilizada para clasificación junto con un grupo de datos etiquetados con sus respectivas clases y otro grupo de datos sin etiquetar. La línea de puntos representa la función que separa y clasifica estos datos. Todo lo que está por sobre la recta será clasificado como Clase 1 (rojo) y todo lo que está por debajo será clasificado como Clase 2 (azul). También puede observarse que de esta manera no se garantiza que la clasificación sea 100% correcta.

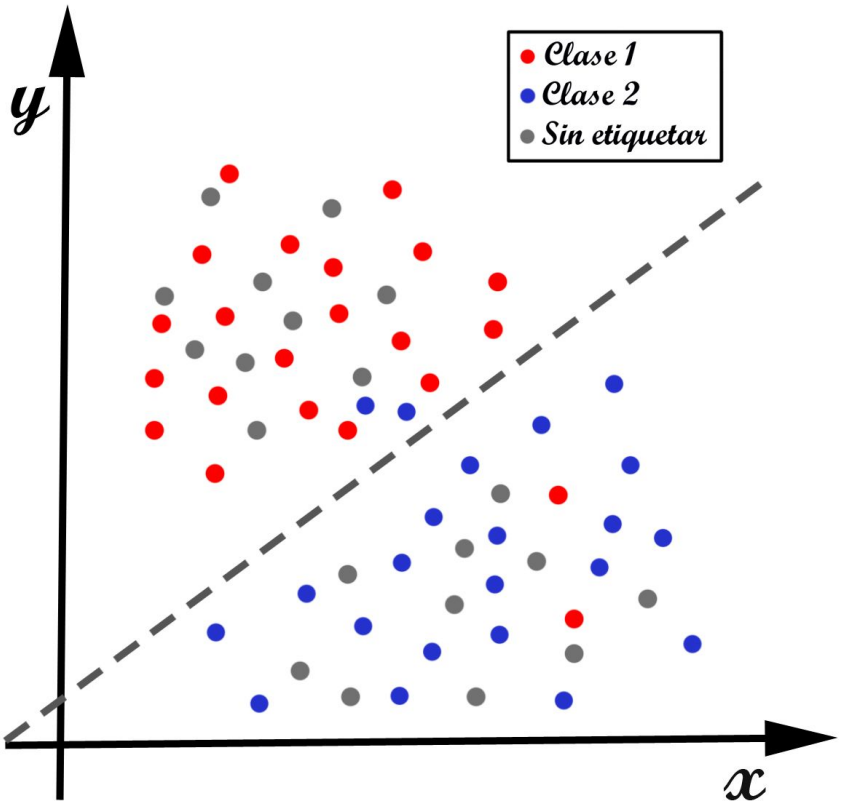

Figura 1: Conjunto de datos y función de clasificación.

2.2.2. Aprendizaje no supervisado

El aprendizaje no supervisado es un método de aprendizaje automático donde se busca generar un modelo de acuerdo a las observaciones. La principal diferencia con respecto al aprendizaje supervisado es que los datos no están etiquetados.

Para obtener el modelo, la información de entrada se organiza y agrupa de acuerdo a ciertas similitudes o diferencias entre los datos. De esta forma se pueden generar clases incluso cuando no fueron provistas anteriormente.

En la actualidad, para el aprendizaje no supervisado se suelen utilizar las redes neuronales. Además, existen otras formas particulares de

aprendizaje no supervisado como por ejemplo el análisis de grupos (clustering en inglés) en donde se pueden encontrar algoritmos como K-means o EM (expectation-maximization en inglés).

En la Figura 2 podemos ver un ejemplo del resultado de clustering de datos mediante aprendizaje no supervisado. Los datos están agrupados de acuerdo a las características que el sistema observó.

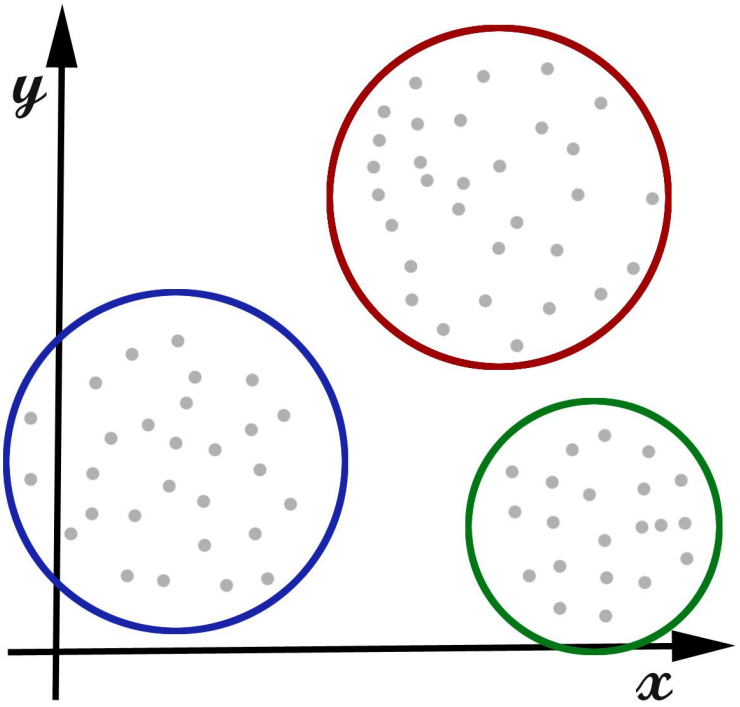

Figura 2: Agrupación de datos no etiquetados mediante aprendizaje no supervisado.

2.2.3. Evaluación

Para medir el desempeño de sistemas de aprendizaje automático se suelen realizar diferentes análisis:

- cuantitativo (métricas, y además usando datos etiquetados)

- cualitativo (análisis de errores)

En esta sección presentamos las principales métricas utilizadas en el problema de clasificación, que serán utilizadas luego en el desarrollo del trabajo.

En un problema de clasificación binario, es decir, con dos resultados posibles, se suelen utilizar las siguientes métricas:

- **Accuracy:** indica la relación entre la cantidad de aciertos sobre el total de datos.
- **Precision:** indica con que certeza un resultado obtenido es correcto.
- **Recall:** indica el porcentaje de cobertura de los elementos que debe encontrar.
- **F-score:** es la media armónica entre precision y recall.

Para esto también es bueno entender los conceptos de:

- **True positives (TP):** el sistema selecciona algo que debe ser seleccionado.
- **True negatives (TN):** el sistema no selecciona algo que no debe ser seleccionado.
- **False positives (FP):** el sistema selecciona algo que no debe ser seleccionado.
- **False negatives (FN):** el sistema no selecciona algo que debe ser seleccionado.

Utilizando estos conceptos, se pueden expresar a las métricas de la siguiente manera:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Por ejemplo, supongamos que tenemos un problema de clasificación de elementos rojos y azules. El objetivo es seleccionar los elementos azules. En la Figura 3, se representa una selección (la circunferencia en el centro) del conjunto de datos y las cuatro posibilidades que se puede dar para cada clasificación de los datos (True positives, True negatives, etc). Se observa que la selección abarca también elementos rojos (afecta precision) y que no abarca todos los azules (afecta recall).

La accuracy no es la única métrica que se utiliza ya que es poco informativa en conjuntos de datos desbalanceados. Supongamos que se tiene un texto con 10 palabras de las cuales hay que corregir sólo una. Si se propone un sistema que no corrija nada en absoluto, tendríamos una accuracy del 90%, ya que acertó 9 de 10 veces que no debía corregir. Un sistema que no hace nada tiene un alto valor de accuracy.

Para estos casos son útiles las métricas de precision y recall. En este ejemplo la recall es cero, ya que todas las correcciones que debían hacerse, no se hicieron.

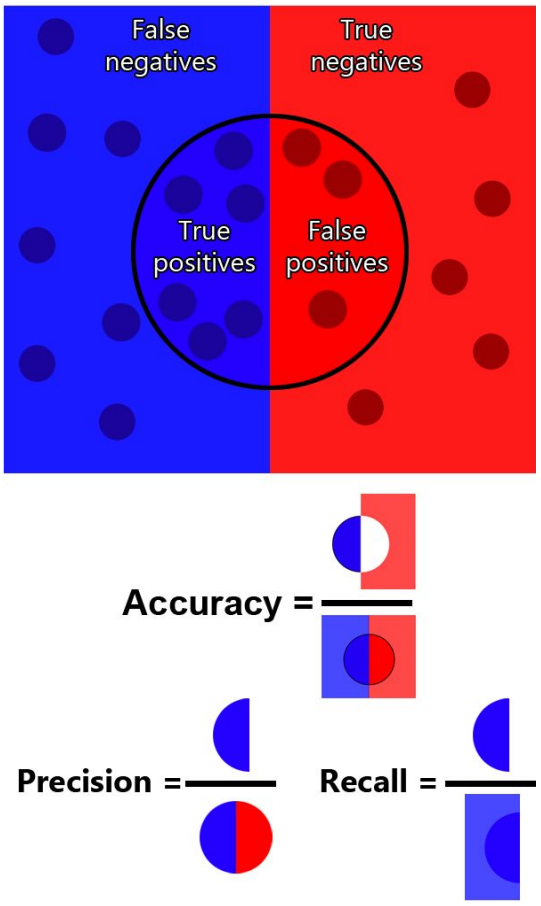

Figura 3: Representación gráfica de las métricas.

Para combinar precision y recall en un solo valor numérico, se utiliza la F-score. Tiene un parámetro que se puede variar para dar más o menos peso a una de las dos medidas. Está definida como:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

El caso más común es en donde tanto precision como recall tienen el mismo peso. Esta medida es conocida como F1-score, debido a que el parámetro β vale 1. La fórmula resultante es:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

2.3. Normalización de Texto

La **normalización de texto** es el proceso de transformación de texto para la obtención de una forma canónica. Usualmente incluye la corrección de errores ortográficos y de tipeo, y la uniformización en el uso de mayúsculas, acentos, signos de puntuación, acrónimos y abreviaciones.

Texto original	Texto normalizado
holaa, queria consultar x él aviso q vi en el diaro d esta mñna	hola, quería consultar por el aviso que vi en el diario de esta mañana

Tabla 1: Ejemplo de normalización de texto.

En la Tabla 1 podemos observar cual sería el resultado de aplicar normalización a un texto dado.

La normalización es especialmente importante a la hora de procesar texto generado por los usuarios en las redes sociales debido a lo altamente ruidosos que son estos contenidos. Es por eso que la normalización además puede contemplar modismos y expresiones propias de las redes.

La reducción del ruido permite mejorar la calidad de la información para su posterior procesamiento. Sin embargo, esto requiere bastante sofisticación lingüística e intuición que muchas veces solo un nativo tiene.

Se puede aplicar en diferentes ocasiones donde se utiliza algún idioma en forma de texto, y donde se requiere que se convierta a una forma apropiada del lenguaje, es decir, texto bien escrito o formado, por ejemplo en aplicaciones de análisis de sentimiento para mejorar el desempeño y los resultados obtenidos.

La normalización de texto presenta ciertas dificultades que no se encuentran en otros dominios. Está evolucionando en todo momento, nuevas palabras, usos y abreviaciones aparecen constantemente. A esto se le agrega el hecho de que se pueden realizar diferentes transformaciones a una misma palabra. Por ejemplo, algunos pueden escribir “porq” y otros “xq” para referirse a “porque”. A su vez, estas transformaciones diferentes pueden ser realizadas por la misma persona.

2.4. Trabajos Previos

Tareas como análisis de sentimiento y text-to-speech han impulsado la normalización de texto, y se utilizan diferentes implementaciones para llevarlo a cabo. Esta área presenta dificultades que no se encuentran en otros dominios ya que el lenguaje utilizado en las redes sociales se encuentra siempre en evolución. Aparecen nuevas abreviaturas, diferentes personas abrevian distinto, lunfardo, etc. La normalización de texto no es un área a la que se le haya dedicado mucho trabajo. Sin embargo, hay áreas bien establecidas como la corrección de errores ortográficos que es algo positivo para esto.

Algunos trabajos previos, como en (Wong et al. 2006), se enfocan en normalizar abreviaturas (“abbreviation expansion”). Usaron traducción automática basada en frases, necesitando un gran corpus, ya que el aprendizaje se hacía a nivel de palabras, donde las palabras no vistas en el entrenamiento eran un problema, y en algunos casos se agregaron correctores ortográficos para mejorar el rendimiento.

Además, introdujeron su sistema ISSAC que trabaja sobre un corrector ortográfico (Aspell[1]) para simultáneamente realizar corrección ortográfica, "abbreviation expansion" y restauración de mayúsculas y minúsculas.

Una línea de investigación en este dominio, ve la normalización como un problema de canal ruidoso ("noisy channel" en inglés). Por ejemplo, (Choudhury et al. 2007) describen un modelo de noisy channel supervisado usando HMMs (Modelos ocultos de Márkov) para normalización de SMS. Luego (Cook & Stevenson 2009) extendieron este trabajo creando un noisy channel no supervisado usando modelos probabilísticos de abreviaciones comunes y eligiendo la palabra de mayor probabilidad.

(Whitelaw et al. 2009) usaron un modelo de noisy channel basado en distancia de edición léxica, generando un conjunto de pares palabra/abreviatura para entrenamiento y sugerencias de errores ortográficos. Cabe destacar que no se enfocaron en texto informal si no en errores ortográficos no intencionales.

(Beaufort et al. 2010) combinaron un modelo de noisy channel con un transductor de estados finitos basado en reglas y obtuvieron buenos resultados, pero solo lo analizaron para SMSs en francés, sin intentarlo para el inglés ni el español.

(Pennell & Liu 2010) y (Pennell & Liu 2011) atacaban las abreviaturas basadas en eliminación (borrar letras) usando modelos estadísticos combinados con un modelo de lenguaje (de ese dominio). (Liu et al. 2011) extendieron el modelo estadístico para que sea independiente del tipo de abreviación. (Liu et al. 2012) también extendieron el modelo para que analice errores ortográficos comunes.

[1] http://aspell.net/

(Han & Baldwin 2011) primero determinaban si dada una palabra fuera de vocabulario (*out-of-vocabulary* o OOV) necesitaba ser expandida o si era otro tipo de OOV bien formada. Para las incorrectas, se generaba un conjunto de posibles candidatos basados en una combinación de distancia de edición léxica y fonética, y luego se elegía el mejor candidato basado en el contexto, usando un modelo de lenguaje e información de dependencia gramatical con parseo. Luego (Han et al. 2012) intentaron construir automáticamente un diccionario de normalización offline usando similitud de distribución de tokens combinado con su distancia de edición.

Técnicas de traducción automática (*machine translation*, MT) entrenadas a nivel de palabras o frases, también son comunes. (Aw et al. 2006) vieron al lenguaje utilizado en SMS como si fuera otro idioma con sus propias palabras y gramática para producir oraciones en inglés gramaticalmente correctas usando MT. (Henríquez & Hernández 2009) entrenaron un sistema de MT utilizando tres diccionarios online de SMS para normalizar mensajes de Twitter. (Kobus et al. 2008) incorporaron una segunda fase en el modelo de traducción que mapea caracteres de las abreviaturas a fonemas, que son vistos como la salida de un sistema de reconocimiento de habla automático (en inglés automatic speech recognition, ASR). Usaron transductores de fonemas no determinísticos para transformar fonemas en palabras del inglés. En el trabajo de (Krawczyk & Raghunathan 2009) se detalla un estudio donde varían el orden del modelo de lenguaje, distorsión de límite y el largo máximo de frase permitido por el sistema MT durante la decodificación. (Contractor et al. 2010) también usan un modelo MT. Para lidiar con el problema de colectar y armar un corpus extenso, crearon automáticamente una lista de pares palabra/abreviatura para entrenamiento usando algunas heurísticas.

(Pennell & Liu 2011) usaron un sistema MT a nivel de caracter. Luego, (Li & Liu 2012) extendieron este modelo para incluir información fonética.

Existen herramientas online que normalizan texto informal y/o mal escrito, que son sólo para el idioma inglés. Algunos ejemplos son transl8it[2] y lingo2word[3]. Fueron lanzados en 2002 y 2001 respectivamente, y esa puede ser la razón por la cual están enfocados principalmente en SMSs, ya que en esa época era un medio muy utilizado y que se caracterizaba por una escritura informal y sintetizada.

Ninguno de los trabajos mencionados anteriormente ataca el problema de normalización de texto en su totalidad. En cambio, intentan solucionar aspectos específicos, como por ejemplo las abreviaciones. Además, son en un mayoría para el inglés, pero ninguno para el español.

Para el idiomas español, muchos trabajos corresponden a los sistemas presentados en el workshop Tweet-Norm 2013 (Alegria et al. 2013) de la SEPLN, o se basan en ellos. A continuación describimos algunos de ellos.

(Ruiz & Cuadros 2013) agregan la técnica de detección de entidades para determinar si el mejor candidato debe ser o no considerado. Luego, en (Ruiz & Cuadros 2014) se presenta un sistema mejorado basado en el que presentaron en la tarea compartida Tweet-Norm 2013 (Alegria et al. 2013).

(Cerón-Guzmán & León-Guzmán 2016), basado en el sistema de (Han & Baldwin 2011), destacan la normalización "uno a muchos" para casos de aglutinación de palabras.

[2] http://www.transl8it.com
[3] https://lingo2word.com

2.5. Corpus y otros recursos existentes

En el taller de Tweet-Norm 2013 (Alegria et al. 2013) se utilizó un corpus que ellos construyeron para realizar la tarea, y que se puede descargar públicamente[4]. Este corpus consiste de un conjunto de tuits correspondiente a los días 1 y 2 de abril de 2013, enviados en el área geográfica de la península ibérica, sin tener en cuenta las regiones que tienen lenguas cooficiales. A partir de ese corpus inicial se generaron dos subconjuntos, uno de desarrollo compuesto por 500 tuits, y otro de test compuesto por 600 tuits.

[4] http://komunitatea.elhuyar.eus/tweet-norm/recursos/#Descargas

3. Construcción de un Corpus de Normalización para el Español de Argentina (Corpus NEA)

Para poder atacar el problema de normalización de texto, se necesitan datos sobre los cuales trabajar. En nuestro trabajo, los datos a utilizar son tweets escritos en español y particularmente, español de Argentina.

Se elige este idioma específico debido a la cercanía y conocimiento que uno tiene sobre el idioma y su utilización en redes. Además, se pueden observar fácilmente muchas particularidades en su uso, como ser el dialecto propio del país, palabras y expresiones en otro idioma (casi en su totalidad, inglés), modificaciones intencionales para reducir el uso de caracteres o el tiempo de escritura, etc.

Otro motivo que nos impulsa a construir un corpus propio es que el corpus existente de la competencia Tweet-Norm 2013 (Alegria et al. 2013) tiene muchas falencias. Por ejemplo, muchas palabras incorrectas aparecen sin corregir.

Construimos un corpus específico del español de Argentina, que consiste de 1200 tweets recolectados desde el 23 de Diciembre de 2017 hasta el 12 de Enero de 2018, cubriendo diferentes horarios del día, y controlando que sean tweets en español publicados en Argentina. Gran parte de estos tweets contienen diferentes problemas de normalización que se deberán solucionar de diversas maneras. Luego de esta recolección, procedimos a corregir manualmente los tweets siguiendo ciertos criterios de corrección

para que haya consistencia en los datos. Nos referiremos a este corpus como “Corpus de Normalización para el Español de Argentina (NEA)”.

En el proceso de anotación, puede ocurrir que ciertas palabras no puedan deducirse por sí solas, y se deba recurrir al contexto para que quien esté corrigiendo sepa cuál es la palabra correcta. Otro caso que puede ocurrir es que una misma palabra tenga diferentes correcciones y esto es también debido al contexto. Y por último, en algunos casos, no se sabe cual es la palabra correcta. En las tablas 2, 3 y 4, vemos ejemplos de estas situaciones.

Tweet	Palabra a corregir	Candidato 1	Candidato 2	Corrección
olaa, que tal?	olaa	ola	hola	hola

Tabla 2: Deducción de corrección por contexto

Tweet	Palabra a corregir	Corrección 1	Corrección 2
xq sos así? xq sí.	xq	Por qué	Porque

Tabla 3: Misma palabra con diferentes correcciones

Tweet	Palabra a corregir	Candidatos
asdfgasd chau	asdfgasd	-

Tabla 4: Desconocimiento de posibles correcciones

3.1. Proceso de Recolección de Tweets

Para obtener tweets, se utilizó una librería de Python llamada Tweepy[5], particularmente utilizando la API de *streaming*, y así obtener tweets en tiempo real. Esta librería permite personalizar la búsqueda de tweets de manera que se pueda filtrar por ubicación (por coordenadas o nombre de ciudad, provincia, país, etc), palabras presentes en el tweet, y otros criterios.

Para este trabajo, el requisito principal es que los tweets sean de Argentina. Para esto, lo que se hizo fue filtrar mediante cuatro coordenadas formando un rectángulo que contenga al país, como se muestra en la Figura 4. Puede observarse que incluye a otros países, y esto no es lo que se busca. Para solucionarlo, una vez que se encuentra un tweet (que cumple con el requisito de ser de una ubicación dentro de ese rectángulo), se analiza si contiene información más específica sobre el lugar de publicación (precisamente país), y que este sea "Argentina". Si no se cumple los requisitos, el tweet se descarta. Por otro lado, se decidió no tener en cuenta los "retweets" ya que son copia de otros tweets y puede que no sean de Argentina.

Al utilizar *streaming*, si los tweets se obtienen, por ejemplo, en horarios de la mañana, la gran mayoría habla sobre un mismo tema, en este caso sobre el comienzo del día (por ejemplo, dar buenos días). La monotonía de temas sobre los que hablan los tweets se quiere evitar, ya que trae poca variedad en las posibles palabras a corregir. Debido a esta situación, se obtuvieron tweets en diferentes horarios del día y además en diferentes días, porque los temas también varían de acuerdo al día de la semana.

[5] http://www.tweepy.org/

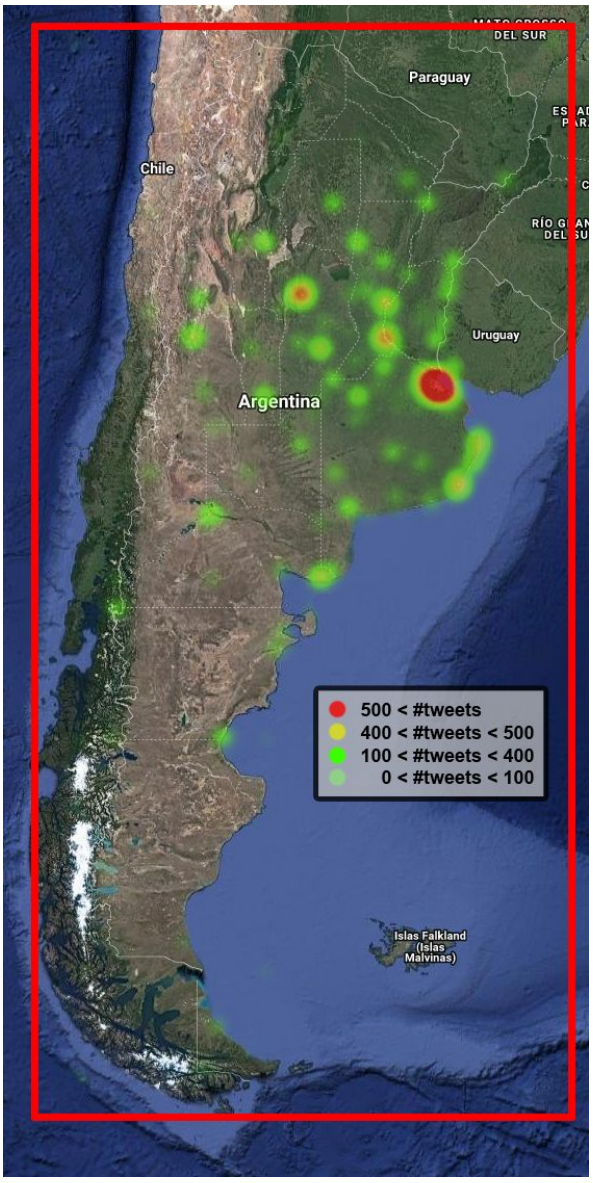

Figura 4: Área de recolección de tweets y distribución representada con mapa de calor.

En la Tabla 5 pueden observarse las diferentes ventanas de tiempo en las que se recolectaron los tweets. Por ventana se recolectaron entre 1000 y 1200 tweets, llegando a un total de 7200 ítems. A su vez en la Figura 4, también se muestra la distribución de la ubicación de los 7200 tweets representada con mapa de calor.

De los tweets recolectados, se hizo un filtrado manual para obtener una muestra representativa y relevante para la tarea de normalización. En el

filtrado, se eliminaron repeticiones de tweets de contenido muy similar o con errores a corregir muy parecidos. También se eliminaron tweets que no contenían lenguaje natural y otros casos especiales. No obstante, se mantuvo al menos un ejemplo de cada tipo de error con el que nos encontramos en la muestra original. Este proceso resultó en la eliminación de 6000 entradas, quedando un corpus final de 1200 tweets.

Bloque	Fecha	Hora de comienzo	Hora de finalización	# Total de tweets	# Tweets resultantes
1	Sábado 23/12/ 2017	19:22:02	19:35:00	1000	312
2	Martes 02/01/ 2018	06:55:42	07:39:12	1000	297
3	Jueves 04/01/ 2018	20:30:48	20:40:54	1000	118
4	Jueves 04/01/ 2018	22:06:42	22:13:56	1000	72
5	Sábado 06/01/ 2018	12:07:16	12:22:15	1000	104
6	Lunes 08/01/ 2018	18:52:31	19:04:40	1000	102
7	Viernes 12/01/ 2018	00:00:00	05:07:08	1200	195

Tabla 5: Cantidades de tweets recolectados y resultantes en las diferentes ventanas de tiempo de recolección.

3.2. Proceso de Anotación Manual

La siguiente etapa fue corregir manualmente todos los tweets recolectados para tener una forma de evaluar el desempeño del sistema. Se debieron identificar las palabras incorrectas e indicar su forma normalizada. Para esto se establecieron criterios de corrección. Una vez establecidos los criterios, se utilizó un editor de texto y se anotaron manualmente las correcciones. Este proceso de anotación manual fue realizado por una sola persona.

3.2.1. Notación de correcciones

La notación de las correcciones consiste en clasificar el error y luego indicar la forma correcta. Se decidió clasificar los errores en dos tipos. Los errores ortográficos (tipo 0) y palabras existentes incorrectas por contexto (tipo 1).

La sintaxis de una corrección será escribir la palabra original, seguida del tipo de error (0 o 1) y luego la forma correcta de la palabra. Es decir:

```
<palabra_original> <tipo_de_error> <forma_correcta>
```

Para aquellos tweets que tengan más de una corrección, se utilizará un separador, el cual será " | ".

El orden de corrección es exactamente el mismo al orden en que aparecen las palabras a corregir en el tweet. Además, si una palabra se repite, se repite la corrección.

En la Tabla 6 se pueden observar ejemplos de la notación utilizada.

Tweet	Correcciones
Hola holaa	holaa 0 hola
havia una ves un perro	havia 0 había | ves 1 vez
El florero se callo del balcon	callo 1 cayó | balcon 0 balcón
yaa yaa yaa	yaa 0 ya | yaa 0 ya | yaa 0 ya

Tabla 6: Ejemplos de notación de correcciones.

3.2.2. Criterios de Corrección

A continuación describimos los criterios de corrección utilizados para la anotación manual. Además se presentan ejemplos para aclarar casos específicos que pueden generar confusión.

Algunos de los errores más comunes a corregir son:

- ausencia o agregado de tildes (ej: jamas, péro)
- errores ortográficos (ej: vurro, bentaniya)
- errores ortográficos que resultan en palabra existente (ej: “cayo” en “me cayo la boca”)
- repetición de caracteres (ej: hoooolaaaa, chaaaaau)
- falta o agregado de letras (ej: saluds, computqadora)
- palabras juntas (ej: porfavor, tequiero)
- abreviaturas, siglas, acrónimos, etc (ej: admin, aprox, EEUU)
- intercambio de sonidos por números (ej: 100pre, dormi2)
- intercambio de letras por números según su forma (ej: h0la, hol4)
- errores de mayúsculas/minúsculas (ej: cAbALLo)
- errores sintácticos (ej: “vamos a las casa”)
- palabras mal escritas en otro idioma (ej: “helloo”, “windou”)

Para la mayoría de los casos mencionados, no se presentan ambigüedades o dudas al momento de corregir. Sin embargo, hay situaciones particulares que requieren de una explicación un poco más extensa y que presentamos a continuación.

Diminutivos y aumentativos: Los diminutivos y aumentativos son considerados palabras correctas. En caso de ser una variante de estos, se normalizan al diminutivo/aumentativo más simple y correcto. En la Tabla 7 se muestran algunos ejemplos de cómo corregir aumentativos y diminutivos.

Palabra original	Corrección
casononón	caserón
chiquititito	chiquito

Tabla 7: Ejemplos de corrección de diminutivos y aumentativos.

Interjecciones: Las interjecciones (ja, aj, oh, eh, etc) se normalizan al mínimo necesario. En la Tabla 8 se muestra como corregir algunas de las interjecciones.

Palabra original	Corrección
aajjajjaaaajajaja	ja
ooooohhhhh	oh
mmmmm	mm
hmmmmm	mm
grrrrrrrrr	grr

Tabla 8: Ejemplos de corrección de interjecciones.

Tildes: Se analiza cuidadosamente el uso de tildes, de acuerdo a las reglas de la lengua castellana. Por ejemplo en *"él sol"*, vemos que la tilde en "él" es incorrecta y la corrección debería ser "el". Se debe tener en cuenta además la posibilidad de que aparezcan tildes invertidas (à, è, etc), que también deben ser corregidas.

Mayúsculas y minúsculas: Las reglas estándar de uso de mayúsculas y minúsculas deben respetarse. En particular, para los nombres propios y también para casos donde el nombre propio original contiene mayúsculas y minúsculas dispuestas de una forma particular. En la Tabla 9 vemos un ejemplo de ese caso. Además se presentan otros ejemplos.

Palabra original	Corrección
whatsapp	WhatsApp
caSA	casa
mabel	Mabel

Tabla 9: Ejemplos de corrección de mayúsculas y minúsculas.

Aglutinación de palabras: Cuando la corrección consiste en más de una palabra, se deben separar mediante guión bajo. En la Tabla 10, se ven ejemplos de cómo corregir la aglutinación de palabras.

Palabra original	Corrección
porfavor	por_favor
tequiero	te_quiero

Tabla 10: Ejemplos de corrección de aglutinación de palabras.

Ambigüedad en correcciones: Para las palabras con correcciones ambiguas, se debe indicar en la corrección las posibles correcciones separadas por barra ("/"). Esto es solo cuando la ambigüedad no es resuelta por el contexto. En la Tabla 11, presentamos ejemplos de correcciones para este caso.

Tweet	Correcciones
me cayo la boca	cayo 1 callo/calló
ayer salí a cormer	cormer 0 comer/correr

Tabla 11: Ejemplos de palabras con más de una posible corrección.

Abreviaturas, siglas, acrónimos, etc: Solo se normalizan a su palabra completa las abreviaturas. Con respecto al resto (siglas, acrónimos, etc), se normalizan de manera que hagan un uso estándar de mayúsculas y puntos. En la Tabla 12 vemos ejemplos de correcciones de abreviaturas y siglas.

Palabra original	Corrección
aprox	aproximadamente
tel	teléfono
EEUU	EE.UU.
fifa	F.I.F.A.

Tabla 12: Ejemplos de correcciones de abreviaturas y siglas.

Normalización desconocida: Para aquellos casos en los que no se sabe la forma normalizada, se pone ' - ' (guión medio) como corrección. En la Tabla 13 mostramos un ejemplo con las correcciones de un tweet, en donde hay una palabra cuya forma estandarizada es desconocida.

Tweet	Correcciones
Holaaaa agsfsdf	Holaaaa 0 Hola | agsfsdf 0 -

Tabla 13: Ejemplo de tweet con correcciones desconocidas.

Lunfardo, argentinismos, jerga de internet, etc: Todo lo que es "lunfardo", argentinismo, jerga de internet y no está mal escrito, se considera correcto. Por ejemplo "bondi" es correcta pero "vondi" no. Para casos como "hdp", "lpm", etc, se normalizan a "HDP", "LPM", etc. Cuando haya un verbo que deba corregirse, la normalización será con voseo verbal. Si en el verbo no aplica voseo pero está bien escrito, no debe corregirse. En la Tabla 14 vemos ejemplos de correcciones de lunfardos y de la aplicación del voseo verbal.

Tweet	Correcciones
hdp no saves nada	hdp 0 HDP | sabes 0 sabés
andaa y coorre el bondi	andaa 0 andá | coorre 0 corré
anda y corre el vondi	vondi 0 bondi

Tabla 14: Ejemplos de correcciones de lunfardos y aplicación de voseo.

Elementos de internet/Twitter: Direcciones de correos electrónicos, enlaces, usuarios, etiquetas, hashtags, etc, no serán normalizados.

Tweet en forma normal: Si no hay correcciones, dejar en blanco. Si el tweet es, por ejemplo, "*Hola loco"*, no hay nada para corregir.

3.3. Dificultades durante la Anotación Manual

Para la tarea de anotación, hay que tener muy en claro las reglas del español, y también tener un amplio vocabulario, debido a que uno se encuentra con ciertas palabras que son desconocidas para el lector, pero no necesariamente son incorrectas.

Al momento de corregir una palabra hay que estar seguro de que deba corregirse y cómo se va a corregir, ya que quizás quien está corrigiendo, por ejemplo, agrega un acento innecesario.

Al margen del error humano por equivocación o por desconocimiento, como confusiones entre “que” y “qué” o “sino” y “si no”, en la corrección manual, puede ocurrir que diferentes personas corrijan de diferente manera una misma palabra.

Una misma palabra puede corregirse de diferentes formas dependiendo del contexto, y a veces aún analizando el contexto no queda muy claro. Un claro ejemplo de esto puede ser lo que ocurre con “xq” que dependiendo de si es pregunta o no, se corrige a “por qué” o “porque”.

En algunos casos, se tiene que analizar muy bien el contexto para decidir cuál es la corrección. Por ejemplo en “no me voy a acerme”. Aunque la palabra más cercana a “acerme” es “hacerme”, la corrección es “hacer”. En otros casos, hay palabras que no serían la mejor de acuerdo al contexto pero así y todo no se puede decir con certeza que es incorrecta. Si tenemos “en casa de herrero, cuchillo de pelo” no tenemos las pruebas suficientes de que “pelo” no sea la palabra que quiere decir.

Teniendo el criterio de corrección, puede ocurrir que no cubra todos los posibles casos con los que el corrector se puede encontrar y por ende se tomen diferentes decisiones.

Además, hay ciertas palabras que no tienen una forma normalizada, ya sea porque no se parece a ninguna palabra del vocabulario o porque es algo desconocido.

Sucede también que hay ciertas palabras en las que no está claro si pertenecen o no al vocabulario, y que generalmente son modismos, lunfardos o palabras muy utilizadas que no son propias del español. Algunos ejemplos pueden ser “bondi”, “tuit”, “finde”, etc.

También puede ocurrir que no se sepa si son más de una palabra juntas o no. Por ejemplo en “piletaah”, no está claro si debe corregirse a “pileta” o a “pileta ah”.

A continuación mostramos algunos ejemplos reales sobre fragmentos de tweets del corpus.

- Fragmento: “me explican que paso?”
    - ¿“paso” es correcta o debe corregirse a “pasó”?

- Fragmento: “me dejo dudando?”
    - ¿“dejo” es correcta o debe corregirse a “dejó”?

- Fragmento: “no tengo ni 2 pesos shoro”
    - ¿“shoro” debe corregirse a “lloro” o “choro”?

- Fragmento: “No creo estar en sus pensamiento”
    - ¿Cuál es la forma correcta? ¿”su pensamiento” o “sus pensamientos”?

Estos ejemplos nos demuestran de que ni siquiera un humano puede corregir esto con total certeza, lo que nos da la pauta de que no es posible alcanzar métricas perfectas en el desempeño de un sistema de normalización.

4. Un Sistema de Normalización Basado en Reglas

En 2013, la Sociedad Española para el Procesamiento de Lenguaje Natural (SEPLN) realizó un taller para la normalización de tweets Tweet-Norm 2013 (Alegria et al. 2013) en el cual se presentaron 13 sistemas que intentaban resolver el problema.

El sistema presentado en este trabajo comenzó basado en uno de ellos, específicamente en el sistema presentado en (Gamallo et al. 2013). Cabe destacar que este sistema logró el segundo mejor desempeño (medido por precisión) detrás del sistema presentado en (Porta & Sancho 2013) de la RAE.

Lo que se extrajo de allí fue la idea de utilizar un conjunto de recursos léxicos para poder decidir cuándo una palabra es conocida o no, la manipulación de palabras para crear candidatos y un modelo de lenguaje para la elección de la palabra corrección propuesta más probable.

El flujo de datos comienza con el conjunto de tweets que son procesados de cierta forma para analizar palabra por palabra y detectar cuáles deben ser corregidas y cuáles no. Una vez que se detectaron todas las palabras a corregir, se procede a generar los candidatos para corregir cada una de las palabras. Los candidatos se generan aplicando transformaciones a las palabras, siendo a veces necesario que se cumplan ciertas reglas para aplicar alguna transformación o no.

Las transformaciones son básicamente agregado, reemplazo o eliminación de caracteres. En algunos casos especiales, la transformación puede ser un reemplazo completo de la palabra.

Luego de que tenemos los candidatos generados, se debe elegir la palabra propuesta como correcta. El proceso de elección se hace teniendo en cuenta las palabras previas y siguiendo un modelo de lenguaje.

Una vez seleccionado el candidato ganador, se pasa a la siguiente palabra para corregir y se realiza el mismo procedimiento.

Cuando se han corregido todas las palabras, se construye el conjunto de tweets con sus correspondientes correcciones que entrega el sistema como salida.

4.1 Arquitectura del sistema

El sistema está dividido cinco componentes principales, de los cuales tres son los que contienen la lógica esencial que corresponden a la etapa de clasificación, generación y elección de palabras. Los dos componentes restantes son los encargados de organizar los datos, tanto para la preparación previa al procesamiento como la organización para su presentación una vez finalizado el proceso de detección y corrección. Los componentes además pueden estar acompañados de componentes más pequeños que aportan al funcionamiento de quien los utiliza. Algunos de ellos no son una parte indispensable del sistema, ya que pueden ser activados o desactivados y algunos otros si forman parte de los recursos necesarios para el funcionamiento del sistema completo. En la Figura 5 se muestra cómo se distribuyen todos los componentes del sistema además de la relación que existe entre ellos y el flujo de datos que recorre el sistema.

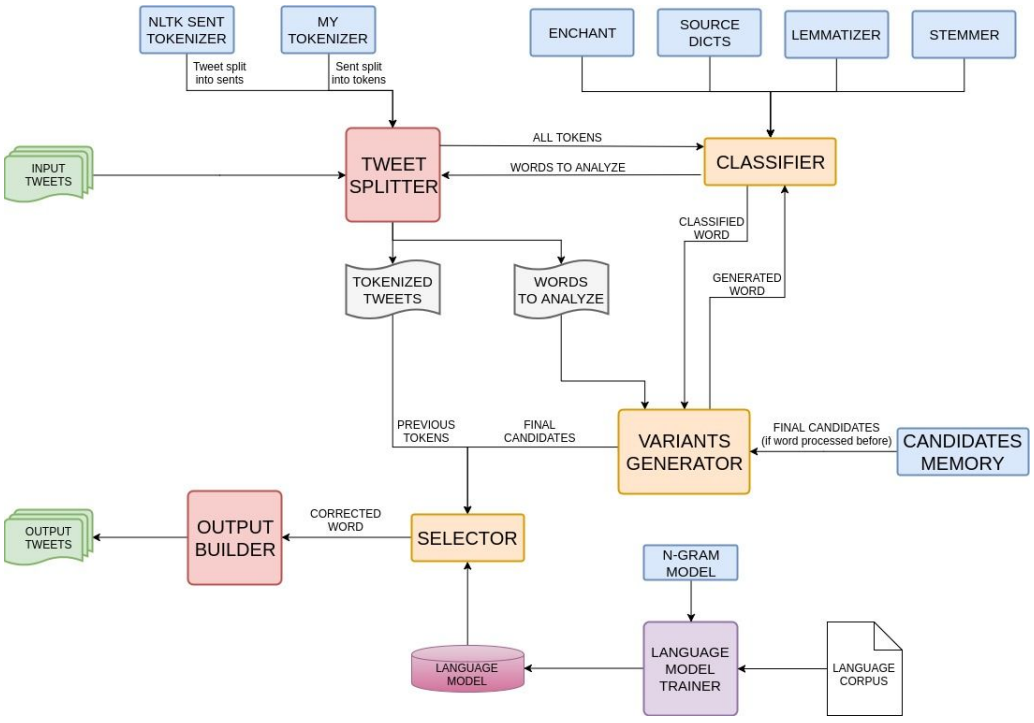

Figura 5: Diagrama de la estructura interna del sistema.

4.2. Preprocesamiento

La primera etapa del flujo de datos en el sistema es la preparación de los tweets que se ingresan para ser procesados.

Cada tweet en el corpus de entrada, debe ser tokenizado para obtener las palabras que se deben analizar. La tokenización está dividida en dos etapas.

Primero, el tweet se separa en oraciones, lo que llamamos (y describimos en la sección 2.1) segmentación de oraciones. A esto se lo realiza utilizando un segmentador de oraciones de una librería de Python que provee

herramientas para el procesamiento del lenguaje natural, llamada nltk[6]. Para esta segmentación se le indica que el lenguaje ingresado es el español.

La segunda etapa es la tokenización de las oraciones, procesamiento mediante el cual obtenemos finalmente cada palabra del tweet. Para esta tarea, utilizamos un tokenizador propio, que es una modificación leve del tokenizador llamado TweetTokenizer que provee la misma librería.

El tokenizador provisto por nltk, sabe distinguir los elementos propios de esta red social, como lo son los hashtags, y elementos de las redes en general, como lo son direcciones de correo electrónico o emoticones. Estos elementos están constituidos por símbolos especiales y son tomados como un solo token.

El tokenizador que definimos, tiene algunas modificaciones para detectar algunas sucesiones de caracteres como un solo token. Por ejemplo expresiones que contienen puntos, como en "A.M.". A estas modificaciones las realizamos agregando expresiones regulares que detecten este tipo de tokens.

Por último, una vez que el tweet está tokenizado, se seleccionan los tokens que se analizarán. Para eso se filtran hashtags, direcciones de correo electrónico, emoticones, números, etc, de manera tal que solo nos quedemos con los tokens útiles. Luego, de esos tokens, se debe detectar cuáles se deben analizar y procesar con más profundidad, mediante el clasificador que se describe en la siguiente sección.

[6] https://www.nltk.org/

4.3. Clasificación

El clasificador es quien se encarga de decidir si una palabra es existente o no. Una vez que los tweets de entrada se encuentran tokenizados, la primer participación del clasificador es en el proceso de detección de las palabras (o tokens) que se deben corregir. Luego la siguiente participación es en la etapa de generación de candidatos, ya que debe controlar que los candidatos que se generan formen parte del idioma (que los clasifique como correctos).

Para clasificar, es decir, para tomar la decisión de si una palabra es correcta o no, se realiza una búsqueda de la palabra en los diferentes recursos léxicos que participan del proceso. Estos recursos son diferentes conjuntos de palabras que forman parte del idioma.

En primer lugar, participa el diccionario de español de Argentina del sistema operativo, al cual accedemos mediante la librería de Python pyenchant[7]. Es la adaptación a python de la librería Enchant. Enchant es una librería que envuelve diferentes librerías y programas de ortografía con una interfaz consistente. Verifica la ortografía de las palabras y sugiere correcciones para aquellas que están mal escritas.

Nos provee una interfaz para poder interactuar mediante funciones con este conjunto de palabras y responder si una palabra ingresada pertenece o no al diccionario. El diccionario contiene la gran mayoría de las palabras del idioma español. De no encontrarse allí, se realiza la búsqueda en diferentes conjuntos o diccionarios de palabras construidos manualmente.

[7] https://github.com/rfk/pyenchant

Estos son:

- Un lemario con palabras del español que no aparecen en el diccionario de sistema, que se armó a partir de un lemario de palabras en español ya armado[8] y luego eliminando las palabras que aparecían en el diccionario del sistema.

- Una lista de nombres propios armado a partir de listas encontradas en wikipedia que además de nombres de personas incluían nombres de organizaciones, compañias, agrupaciones, etc.

- Una lista de expresiones utilizadas en redes sociales a la par de su escritura correcta en español (e.g xq:porque) tomando como base diccionarios de chat existentes (por ejemplo, el diccionario xat Aló! EntelPCS).

- Una lista de lunfardo utilizado en Argentina[9] el cual incluye una amplia variedad expresiones populares en Argentina, dichos, palabras específicas de una jerga y palabras provenientes del extranjero utilizadas en Argentina popularmente.

Además, está la opción de la lematización de la palabra. Se obtiene el lema de la palabra mediante el lematizador que provee la librería de Python spacy[10]. Luego se busca ese lema en los diferentes recursos, y si aparece en alguno, se toma a la palabra original como correcta. Esto es de utilidad principalmente para las palabras en plural y verbos conjugados.

Otro procesamiento opcional es el stemming. Se obtiene la raíz de la palabra mediante el stemmer que provee la librería de Python nltk. Luego se busca que esa raíz sea el comienzo de alguna de las palabras presentes

[8] http://olea.org/proyectos/lemarios/, https://github.com/olea/lemarios
[9] http://www.todotango.com/comunidad/lunfardo/
[10] https://spacy.io/

en los distintos recursos. Si aparece en alguno, se toma a la palabra original como correcta.

Si se encuentra la palabra, la clasificamos como correcta para el idioma español. De no ser esa la situación, debemos ver si es una palabra correcta en otro idioma, en el sistema sólo contemplamos el idioma inglés, utilizando el diccionario de inglés de sistema al cual accedemos con la misma interfaz que nos provee pyenchant.

Por último, si no se cumple esa condición, decimos que es una palabra incorrecta. Al concluir esto, la palabra debe corregirse (para el caso de detección) o no es un candidato válido (para el caso de generación de candidatos).

4.4. Generación de candidatos

La generación de candidatos está basada en reglas y no requiere de ningún entrenamiento, sino que dadas ciertas condiciones genera los candidatos posibles. Algunas de estas reglas fueron las que se utilizaron en (Gamallo et al. 2013), específicamente las de eliminación de repeticiones, corrección de confusiones comunes en el español ("b" por "v", "c" por "s", etc) y palabras con cierta distancia de edición respecto a la palabra original. Estas reglas y las demás que se desarrollaron, son explicadas a continuación.

El sistema puede generar candidatos primarios y secundarios. Esta división se realiza porque los candidatos primarios son generados de acuerdo a cierta lógica y los candidatos secundarios son generados en mayor cantidad y probando diferentes cambios en la palabra que pueden no tener sentido.

La generación de candidatos se realiza para cada palabra que debe corregirse, para luego obtener el candidato ganador que se propone como corrección de la palabra. Los candidatos se generan a partir de aplicarle cambios y transformaciones a la palabra ingresada para corregir.

Los candidatos primarios se generan de diferentes formas. Cada una de estas formas trata de resolver diferentes problemas. Primero se ataca el problema de la repetición de caracteres, en donde se generan variantes de la palabra ingresada. Las variantes generadas son:

- Todos los caracteres repetidos ocurren a lo sumo dos veces (adyacentemente)
    - Ejemplo: ooolllaaa → oollaa
- A lo sumo dos caracteres repetidos (repetición adyacente).
    - Ejemplo: ooolllaaa → oolla, ollaa, oolaa
- A lo sumo un carácter repetido (repetición adyacente).
    - Ejemplo: ooolllaaa → olla, oola, olaa
- Sin repetición de caracteres.
    - Ejemplo: ooolllaaa → ola

De esta forma, se puede corregir cuando en una palabra se repiten caracteres para poner énfasis, o cuando simplemente alguna letra está repetida por error, pero sin perder la situación en donde la palabra correcta deba tener letras repetidas adyacentes como es en el caso de doble “l”, doble “c”, doble “o”, doble “r”, etc.

Para el caso de la repetición, también se busca algún patrón repetido a lo largo de toda la palabra, para detectar situaciones que generalmente son risas. Ejemplos: jajajajaja → ja, jejejeje → je.

Luego de esto, se busca corregir errores relacionados al uso o no de tildes. Lo que se hace para este caso es eliminar todas las tildes que ocurren en la palabra ingresada, y generar candidatos que tengan a lo sumo una tilde a la

vez. Estos cambios sólo se generan para las vocales. Por ejemplo para la palabra “canción” se genera “cancion”, “cáncion”, “cancíon” y “canción” (además de los candidatos que se general al atacar los demás problemas).

Otro problema es la aglutinación de palabras, es decir, las palabras escritas juntas sin ser separadas por espacios. El sistema, por defecto, detecta solo hasta dos palabras juntas, ya que es muy costoso probar todas las combinaciones posibles que pueden darse. Debe fijarse si cada parte resultante de la separación en partes de la palabra es perteneciente al idioma. Ejemplo: todobien → todo_bien.

Después se busca solucionar el problema de errores ortográficos comunes. Los errores comunes que ataca son confusiones entre “v” y “b”, entre “c”, “z” y “s”, entre “g” y “j”, etc. Estas reglas de corrección se encuentran en un diccionario de errores comunes. Las correcciones se realizan manteniendo la pronunciación de la palabra, es decir, para un error común como es la confusión entre “c” y “s”, si ocurre “sa” dentro de la palabra a corregir, la letra “s” podrá cambiarse por una “z” pero no por una “c”.

Otra transformación que se le realiza a las palabras es el cambio de mayúsculas o minúsculas, generando dos posibilidades diferentes, que son, todo minúscula o solo la primer letra en mayúscula (para nombre propios o comienzos de oración).

Todas estas transformaciones y operaciones que se realizan sobre las palabras, son de manera acumulativa, es decir, al ingresar una palabra primero se le realizan las transformaciones relacionadas a la repetición de caracteres, luego pueden corregirse errores de acentuación, después corregir otros errores ortográficos y además errores en mayúsculas o minúsculas. De esta forma podremos generar el candidato “Pablo” cuando la palabra a corregir era “paaavló”.

Finalmente, una vez que se han generado todos los candidatos primarios posibles, se realiza una limpieza para dejar solo aquellos candidatos que sean palabras existentes en el idioma.

El filtro se realiza utilizando el clasificador de palabras y solamente dejando las palabras que resultan correctas utilizando los criterios descritos en la subsección anterior.

Todas estas técnicas son las que generan los candidatos primarios. Los candidatos secundarios son generados sólo si no hay candidatos primarios que pertenezcan al lenguaje, es decir, que luego de generar todas las posibilidades, hayan quedado palabras que pasaron el filtro de correctitud.

Los candidatos secundarios son los que se generan a partir de realizarle tres tipos diferentes de operaciones a la palabra ingresada. Estas operaciones son, agregado, reemplazo y eliminación de caracteres. La generación de candidatos secundarios no sigue reglas ya que genera palabras como resultado de eliminar una letra a la vez, de agregar una letra a la vez o reemplazar una letra a la vez. Este agregado y reemplazo de caracteres se realiza probando con todas las letras del abecedario.

Por último, se realiza el mismo filtro que se le realiza a los candidatos primarios, para que solo queden como candidatos palabras correspondientes al idioma.

En el filtro de palabras es donde entra en juego el diccionario de expresiones propias de redes sociales o de mensajería, para así poder agregar a los candidatos transformaciones casi totales de las palabras. Ejemplo: xq → porque, tqm → te_quiero_mucho.

Las palabras resultantes que fueron obtenidas de ese diccionario no deben pasar por el filtro del idioma ya que tienen como requisito ser parte del lenguaje para constituir este diccionario.

El generador de candidatos, va acompañado de otro componente que juega el rol de memoria. Dado que los candidatos que se generan dependen solamente de la palabra ingresada para corregirse y no dependen de ningún otro aspecto, lo que se hace es buscar en esta memoria si esa palabra ya ha sido vista, y si ese es el caso, se obtienen los candidatos que se generaron para esa palabra. Si todavía no se ha visto, se procede a la generación normal de candidatos y se guarda este conjunto de palabras obtenidas para estar listas en caso de una próxima aparición de la misma palabra a corregir.

4.5. Selección de candidatos

Una vez que se tienen los candidatos generados para una palabra a corregir, se procede a la selección de la mejor opción, es decir, la palabra que se propondrá como correcta.

Para esta etapa, es necesario tener un modelo de lenguaje del idioma español, ya que la decisión se toma de acuerdo al contexto de la palabra que debe corregirse.

El modelo está basado en n-gramas, donde con n-grama se refiere a una tupla de n palabras ($palabra_1$, $palabra_2$, …, $palabra_n$), en este caso, contiguas, con lo que se calculará la probabilidad de que ocurra $palabra_n$ dado que ocurren las n-1 palabras anteriores.

El modelo de n-gramas es obtenido mediante entrenamiento no supervisado. Particularmente, para calcularlo, se utiliza Back-Off, un

método de suavizado (en inglés, smoothing) de n-gramas para lidiar con aquellas palabras que no se ven o se observan poco durante el entrenamiento.

Las probabilidades se obtienen mediante estimación por máxima similitud (maximum likelihood estimation, en inglés) calculando la relación entre conteos de frecuencia. Por ejemplo para el caso de n=3, se calcula la cantidad de ocurrencias del trigrama ($palabra_1$, $palabra_2$, $palabra_3$), sobre la cantidad de ocurrencias del bigrama ($palabra_1$, $palabra_2$).

La fórmula que representa la probabilidad de que ocurra una palabra ($palabra_3$) dadas las dos anteriores ($palabra_1$, $palabra_2$) es igual a:

$$P(palabra_3 \mid palabra_1, palabra_2) = \frac{count(palabra_1, palabra_2, palabra_3)}{count(palabra_1, palabra_2)}$$

a donde count es la cantidad de ocurrencias de los n-gramas.

La palabra que debe corregirse, ingresa junto con el conjunto de candidatos generados y se realiza la búsqueda del contexto de la palabra (las dos palabras anteriores). Una vez hecho esto, se obtienen las probabilidades (previamente calculadas, durante la etapa de entrenamiento del modelo) para cada candidato dado que ocurren las dos palabras anteriores. Finalmente, se toma como candidato ganador a aquel que participa en el 3-grama que tiene la mayor probabilidad de ocurrencia.

5. Experimentos

En esta sección se presentan los experimentos realizados junto con los resultados obtenidos y un análisis de errores correspondiente a cada resultado.

Como los sistemas a evaluar no utilizan datos anotados de entrenamiento, se utiliza la totalidad del corpus como corpus de evaluación.

5.1. Evaluación

Para este problema las métricas que se utilizarán son accuracy, precision, recall y F1 para dos aspectos a evaluar.

El primer aspecto es lo que llamamos “detección de palabras a corregir”. Esto es, del conjunto total de “tokens” que contienen los tweets, detectar cuáles de ellos deben ser corregidos. Un acierto es una palabra detectada y que debía detectarse (acierto positivo), o una palabra no detectada y que no debía detectarse (acierto negativo).

De esta forma, D-accuracy es la relación entre la cantidad de estos aciertos y la cantidad total de palabras que se analizan.

D-precision es la cantidad de aciertos positivos sobre la cantidad de palabras que el sistema detecta (ya que puede haber detectado palabras que no debían detectarse). Con esto, se penaliza la detección errónea y nos responde ¿Que tan confiable es el sistema cuando detecta una palabra?

D-recall es la cantidad de aciertos positivos sobre la cantidad de palabras que el sistema debería haber detectado (ya que puede ocurrir que haya pasado por alto algunas palabras). Con esto se penaliza la falta de detección y nos responde ¿Cuánto el sistema está detectando lo que se busca?.

El segundo aspecto es lo que llamamos "corrección de palabras". Esto es, del conjunto total de palabras, detectarlas y además corregirlas correctamente. Este aspecto es más exigente que el anterior, pero penaliza de la misma manera las palabras no detectadas y las corregidas incorrectamente. Un acierto es una palabra que debía corregirse y fue corregida correctamente (acierto positivo), o una palabra no detectada y que no debía detectarse (acierto negativo).

De esta forma, C-accuracy es la relación entre la cantidad de estos aciertos y la cantidad total de palabras que se analizan.

C-precision es la cantidad de aciertos positivos sobre la cantidad de palabras que el sistema corrigió (ya que puede haber corregido palabras que no debía o corregido mal). Con esto, se penaliza la corrección errónea y nos responde ¿Que tan confiable es el sistema cuando corrige una palabra?.

A su vez, C-recall es la cantidad de aciertos positivos sobre la cantidad de palabras que el sistema debería haber corregido (ya que puede ocurrir que haya pasado por alto ciertas palabras que debía corregir o corregido mal). Con esto se penaliza la falta de corrección y nos responde ¿Que tanto está corrigiendo el sistema lo que debe ser corregido?.

Finalmente, se utilizará D-F1 y C-F1 para la F1-score de detección y F1-score de corrección respectivamente.

Recordemos que el corpus puede incluir dos diferentes tipos de errores. Errores de tipo 0 y errores de tipo 1 como lo explicado en la sección 3.2.1. Ya que el sistema no corrige errores de tipo 1, no se tienen en cuenta al momento de medir el desempeño del sistema. Además, no se tienen en cuenta las mayúsculas o minúsculas (lo que en inglés se dice case insensitive) para evaluar la corrección. Puede ser que una palabra esté bien corregida ortográficamente pero que haya un error en mayúsculas o minúsculas. Esto no se penaliza.

En la Tabla 15 vemos un ejemplo en donde son calculadas las diferentes métricas de desempeño.

Tweet	No deves Aser leña del arbol caiido
Correcciones esperadas	deves 0 debes | Aser 0 hacer | arbol 0 árbol | caiido 0 caido
Correcciones propuestas	Aser 0 Hacer | leña 0 - | arbol 0 arboleda
D-accuracy	4/7 = 57.14%
D-precision	2/3 = 66.67%
D-recall	2/4 = 50%
D-F1	(2*(2/3)*(2/4)) / ((2/3)+(2/4)) = 57.14%
C-accuracy	3/7 = 42.86%
C-precision	1/3 = 33.33%
C-recall	1/4 = 25%
C-F1	(2*(1/3)*(1/4)) / ((1/3)+(1/4)) = 28.57%

Tabla 15: Ejemplo de métricas calculadas para un tweet.

5.2. Sistema Baseline

Un sistema baseline es un sistema con los componentes mínimos para su funcionamiento y cuyo desempeño es tomado como punto de referencia para delimitar un límite inferior en las métricas.

Nuestro baseline es un normalizador de texto out-of-the-box, el cual cumple la función de detección y corrección ortográfica. Está basado completamente en el uso de la librería de python pyenchant**,** utilizando el diccionario de español de Argentina (código es_AR).

En la interfaz de la librería se encuentra una función que detecta si una palabra es correcta o no, utilizando el diccionario antes mencionado. Este método cumple el rol de la detección de las palabras a corregir.

La generación de candidatos, se realiza mediante una función de la librería que, dada una palabra de entrada, sugiere las formas correctas para esa palabra. Para la selección del candidato ganador, se elige el primer candidato que está en la lista de candidatos generados.

5.3. Configuración Inicial

Para los experimentos preliminares con nuestro sistema, utilizaremos una configuración básica que no utiliza stemming ni lematización. Además, utilizaremos un modelo de lenguaje de 3-gramas entrenado con el corpus Ancora. Luego, estudiaremos algunas variantes de estos parámetros. Al sistema normalizador desarrollado lo llamaremos “Twnorm” y así será referenciado en las tablas que se presentarán a continuación.

5.4. Experimentos con el corpus Tweet-norm

En este experimento utilizamos el corpus provisto para la competencia Tweet-Norm 2013 (Alegria et al. 2013). Además se utiliza la configuración inicial descrita anteriormente.

En las tablas 16 y 17 pueden verse los valores de las métricas de detección y corrección del normalizador desarrollado junto con los valores arrojados por el normalizador baseline.

	Detección			
	D-Acc	**D-Prec**	**D-Rec**	**D-F1**
Baseline	94.08%	55.64%	95.33%	70.27%
Twnorm	94.73%	59.52%	87.50%	70.85%

Tabla 16: Resultados de la evaluación en detección para los sistemas de normalización sobre el corpus Tweet-norm.

	Corrección			
	C-Acc	**C-Prec**	**C-Rec**	**C-F1**
Baseline	89.31%	18.29%	31.33%	23.10%
Twnorm	92.33%	37.64%	55.33%	44.80%

Tabla 17: Resultados de la evaluación en corrección para los sistemas de normalización sobre el corpus Tweet-norm.

5.4.1. Análisis de resultados

Para la etapa de detección lo que llama la atención es la baja precisión que tiene el sistema. Esto es porque el normalizador detecta mucho más de lo que se espera en el corpus anotado.

Al revisar las palabras detectadas que según el corpus no debían ser corregidas, vimos que eran palabras realmente incorrectas, es decir, estaban bien detectadas pero faltaban las anotaciones de las correcciones en el corpus.

Este es el principal motivo que nos impulsó a generar un corpus propio.

Por otra parte se puede observar en el corpus, la conjugación de los verbos correspondientes al uso del "vosotros". Para esos casos, el normalizador, no genera los candidatos correspondientes.

5.5. Experimentos con Corpus NEA

En estos experimentos utilizamos el corpus que construimos para evaluar nuestro sistema de normalización utilizando diferentes configuraciones.

5.5.1. Lematización o Stemming

En esta sección analizamos el impacto del lematizador y el stemmer descritos en la sección 4.3.

En las tablas 18 y 19 se muestran los resultados obtenidos para detección y corrección luego de la ejecución del normalizador con las configuraciones correspondientes.

Se puede ver que con el lematizador activado, el sistema se desempeña mejor que sin activarlo o con el stemmer activado. Sin embargo, el costo de tiempo es mucho mayor. Utilizando el lematizador, la etapa de detección tarda 3 veces más y la etapa de generación de candidatos, 185 veces más. Este gran consumo de tiempo no equipara la pequeña mejora en el desempeño del sistema.

Por otro lado, el desempeño con stemming disminuye bastante. Esto es debido a que considera correcta a aquellas palabras cuyo stem sea el comienzo de alguna palabra que aparezca en alguno de los recursos literarios. De esta forma se tienen muchos falsos negativos, lo que provoca una baja recall.

Luego, tomamos la decisión de utilizar el normalizador sin lematización ni stemming.

	Detección			
	D-Acc	**D-Prec**	**D-Rec**	**D-F1**
Twnorm	96.97%	80.03%	84.11%	82.02%
Twnorm+Lem	97.00%	80.33%	83.97%	82.11%
Twnorm+stem	94.89%	80.57%	51.77%	63.04%

Tabla 18: Resultados de la evaluación en detección para los sistemas de normalización sobre nuestro corpus utilizando diferentes configuraciones.

	Corrección			
	C-Acc	**C-Prec**	**C-Rec**	**C-F1**
Twnorm	94.22%	49.53%	52.06%	50.76%
Twnorm+Lem	94.26%	49.80%	52.06%	50.90%
Twnorm+stem	92.76%	42.05%	27.02%	32.90%

Tabla 19: Resultados de la evaluación en corrección para los sistemas de normalización sobre nuestro corpus utilizando diferentes configuraciones.

5.5.2. Modelos de Lenguaje

En esta sección analizamos distintas configuraciones para el modelo de lenguaje utilizado en la selección de candidatos (sección 4.1.3).

Nos concentramos en dos aspectos de los modelos: el corpus de entrenamiento, y el valor del parámetro n (tamaño de la ventana de palabras).

En cuanto al corpus, probamos dos corpus de acceso público. Uno de los corpus utilizado es el de AnCora[11] en español. Consiste de diferentes escritos en idioma español y principalmente textos periodísticos. El otro corpus es una muestra gratuita del corpus del español creado por Mark Davies (Davies 1999). Esta muestra contiene 2 millones de palabras.

Para el tamaño de los n-gramas, probamos con valores de n en {2, 3, 4}.

[11] http://clic.ub.edu/corpus/es/ancora

En la Tabla 20 se muestran los valores de las métricas de corrección sobre el corpus NEA para todas las combinaciones. Las métricas de detección no son mostradas aquí ya que el modelo de lenguaje no participa en esa etapa.

Se puede ver que el cambio del parámetro n no afecta los valores. Los resultados para bigramas y trigramas difieren sólo en dos ítems, mientras que para 4-gramas se dan exactamente las mismas correcciones que para trigramas.

Visto que los resultados son casi iguales, tomamos la decisión de usar n=3, ya que consideramos intuitivamente que mirar sólo la palabra anterior (n=2) puede en ocasiones no ser contexto suficiente.

Con respecto a los diferentes corpus, tampoco se observa una variación significativa. Si bien con el Corpus del Español (BYU) se obtienen mejores resultados tanto en detección como corrección, en ninguno de los casos esas mejoras superan el 1%.

n	AnCora3				Corpus del Español (BYU)			
	C-Acc	C-Prec	C-Rec	C-F1	C-Acc	C-Prec	C-Rec	C-F1
2	94.22%	49.53%	52.06%	50.76%	94.28%	50.20%	52.77%	51.45%
3	94.22%	49.53%	52.06%	50.76%	94.28%	50.20%	52.77%	51.45%
4	94.22%	49.53%	52.06%	50.76%	94.28%	50.20%	52.77%	51.45%

Tabla 20: Resultados de la evaluación en detección y corrección para los sistemas de normalización utilizando diferentes modelos de lenguaje.

5.5.3. Resultados Finales

En esta sección comparamos nuestro normalizador con el baseline, siempre usando la configuración elegida de acuerdo a los resultados de las secciones anteriores.

En la tabla 21 y 22 mostramos los valores de las métricas de detección y corrección del sistema baseline y nuestro normalizador con la configuración establecida.

Se puede ver que nuestro normalizador supera al baseline en todas las métricas excepto en recall de detección. Esto nos quiere decir que nuestro normalizador detecta menos palabras que el baseline. Dicho de otra manera, considera correctas más palabras. El problema es que no son correctas en realidad. Se debe a que los recursos agregados aparte del diccionario del sistema, contienen palabras incorrectas. Puede solucionarse eliminando a mano las palabras incorrectas presentes en estos recursos.

La diferencia de precision en detección se debe a que nuestro normalizador agrega más palabras al lenguaje, como por ejemplo los lunfardos. Son detectados como incorrectos usando el baseline, mientras que nuestro normalizador sabe que son correctos, aumentando así la precisión. Podría mejorarse aún más, agregando nuevas palabras a los recursos y de esta forma conseguir que el sistema no diga que una palabra es incorrecta cuando en realidad no lo es. Hay que destacar que la baja D-precision del sistema baseline impacta directamente en la C-precision, ya que termina corrigiendo palabras que no debe.

Si miramos a los valores de la corrección, el aumento en los valores de las métricas es mucho más notable. Se debe principalmente a que la generación de candidatos es mucho más amplia, destacando la transformación total de palabras que utilizan el diccionario de expresiones de redes sociales, candidatos que no son generados en el sistema baseline.

Además la elección del candidato con un criterio razonable utilizando el modelo de lenguaje y no seleccionando el primer candidato por orden alfabético como en el caso del sistema baseline, aporta a que los valores de corrección mejoren ampliamente.

	Detección			
	D-Acc	**D-Prec**	**D-Rec**	**D-F1**
Baseline	96.16%	71.29%	89.65%	79.42%
Twnorm	96.97%	80.03%	84.11%	82.02%

Tabla 21: Resultados de la evaluación en detección para el sistema baseline y el sistema de normalización con los parámetros elegidos.

	Corrección			
	C-Acc	**C-Prec**	**C-Rec**	**C-F1**
Baseline	90.46%	18.56%	23.33%	20.67%
Twnorm	94.22%	49.53%	52.06%	50.76%

Tabla 22: Resultados de la evaluación en corrección para el sistema baseline y el sistema de normalización con los parámetros elegidos.

5.5.4. Análisis de Errores

En esta sección vemos las diferentes fuentes de error tanto para la detección como para la corrección.

Si bien los resultados del desempeño de la detección de palabras a corregir es bueno, se puede mejorar.

Vimos que hay palabras existentes en los diferentes recursos, que no son palabras correctas y que no deben ser pasadas por alto. Aunque requiere mucho esfuerzo, esto es simple de solucionar, simplemente haciendo una limpieza de los recursos utilizados. Algunos casos de las palabras que aparecen incorrectamente son “hna”, “hambree” y “sols”.

Por otro lado, hay palabras existentes que no están en los recursos y por lo tanto se las detecta como incorrectas. Algunos casos son “Messi”, “birra” y “pibes”. Aumentando el universo de palabras en los recursos, esto se mejoraría.

Recordemos que para la etapa de corrección, se penaliza también errores en la detección, ya que si una palabra que debe ser corregida no es

detectada, o si se detecta como incorrecta una palabra correcta, se toman como mal corregidas.

Para analizar la corrección dejaremos de lado esos casos y nos centraremos en las palabras corregidas por el sistema y que se esperaba que fueran corregidas.

Del total de palabras detectadas y que debían ser corregidas, pero no se corrigieron correctamente:

- un 21% tienen una corrección que no es considerada una palabra correcta. Es decir, no aparece en el diccionario de sistema ni el resto de recursos, y por mas que se genere el candidato, será filtrado. Agregar palabras a los recursos esto se mejora.

- un 12% tienen una corrección que es considerada existente pero no es generada en el proceso de generación de candidatos. Se puede mejorar agregando heurísticas de generación de candidatos.

- un 6% es por error de elección del candidato. Para mejorar esto hay que trabajar sobre el modelo de lenguaje y el método de selección.

Un fallo común del normalizador es cuando debe separar palabras. Al encontrarse con tokens como “enserio”, “lavida”, “nosé”, no es capaz de generar los tokens “en_serio”, “la_vida” y “no_sé”.

El normalizador tampoco resuelve ambigüedades. Es decir para tokens como “novixs” y “emplead@” no genera “novios/as” y “empleado/a”.

Otro error que se ve con frecuencia es la normalización utilizando el voseo verbal. En la anotación manual se espera que el normalizador lo corrija con ese acento, pero el sistema real no funciona así. Para casos como “haganse”

y “traguemonos” se espera “hagansé” y “traguemonós” y no “háganse” y “traguémonos”, que es como el sistema los corrige actualmente.

Finalmente, algunos problemas en la detección o corrección pueden ser causados por errores en la corrección manual, provocados por palabras que debían ser corregidas y se las pasó por alto, y por correcciones mal hechas, incluyendo los casos en que se corrigen palabras que no debían ser corregidas.

6. Conclusiones y Trabajo Futuro

En el presente trabajo se busca solucionar el problema de datos ruidosos mediante la normalización de texto. En nuestro caso, para los datos ruidosos construimos un corpus con tweets escritos en español de Argentina. Además desarrollamos un sistema normalizador de texto sobre el cual realizamos diferentes experimentos y análisis.

El corpus construido incluye las particularidades del español de Argentina, como por ejemplo el uso de lunfardos y voseo. Al momento de realizar el filtrado manual, se prestó atención a no eliminar ninguno de los tipos de errores presentes, y también su pudo mantener la variedad de errores pueden darse. Cubre una gran cantidad de tipos de errores, y varios se repiten en gran cantidad. Comparando con el corpus existente de la competencia de SEPLN, vemos que la anotación realizada en nuestro corpus, fue mucho más completa y detallista. Una causa es que los estándares de corrección eran más explícitos y claros respecto de lo que significa la normalización. No conocemos ningún trabajo previo publicado al respecto.

En los experimentos realizados con el normalizador desarrollado, vemos casi todas las métricas son superiores a las obtenidas con el baseline y algunas con amplio margen. Se destaca la incorporación de palabras propias del dialecto argentino, por lo que pasan a formar parte del lenguaje que maneja el sistema y no son tomadas como palabras incorrectas.

Si bien presenta algunas falencias, como la ausencia de palabras en los recursos o errores en la elección de candidatos, funciona bien corrigiendo palabras muy deformadas y modismos de redes, ya que se aplican

numerosas transformaciones y componentes secuencialmente. Puede que algunos errores tengan su origen en los recursos que aportan a los componentes principales del sistema, pero la estructura modular y con muchos componentes tiene sus ventajas. Si los recursos externos mejoran, automáticamente el desempeño del sistema lo hace. La configuración resulta más sencilla y sectorizada, incluyendo la activación o desactivación de funcionalidades específicas como la lematización. Con la configuración por defecto, con la que se vió que obtiene buenos resultados, el sistema trabaja muy rápido.

A continuación nos centraremos en algunas limitaciones de este trabajo que podrán ser subsanadas en trabajos futuros.

En este trabajo, la solución planteada no utiliza las estructuras sintácticas del lenguaje utilizado en las redes sociales. Dejamos como trabajo futuro un posible análisis de los árboles sintácticos de estas estructuras.

Dos tipos de errores sobre los cuales debe trabajarse más es en la aglutinación de palabras y en la corrección de palabras existentes incorrectas por contexto. La aplicación de los árboles sintácticos mencionados en el párrafo anterior podrían ser de ayuda para esto.

Por otro lado puede utilizarse una diferente selección de candidatos, en donde el contexto no sean solo las palabras anteriores, si no las posteriores también. Para esto se podría utilizar una métrica de certeza en la corrección en relación a la cantidad de candidatos generados, y corregir el texto en varias iteraciones.

7. Bibliografía

Alegria, I. et al., 2013. Introducción a la Tarea Compartida Tweet-Norm 2013: Normalización Léxica de Tuits en Español. *Tweet-Norm*. Available at: http://nlp.lsi.upc.edu/publications/papers/tweetnorm13.pdf.

Aw, A. et al., 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics, pp. 33–40.

Beaufort, R. et al., 2010. A Hybrid Rule/Model-based Finite-state Framework for Normalizing SMS Messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. ACL '10. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 770–779.

Cerón-Guzmán, J.A. & León-Guzmán, E., 2016. Lexical Normalization of Spanish Tweets. In *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion*. Available at: http://dx.doi.org/10.1145/2872518.2890558.

Choudhury, M. et al., 2007. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3), pp.157–174.

Contractor, D., Faruquie, T.A. & Subramaniam, L.V., 2010. Unsupervised Cleansing of Noisy Text. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. COLING '10. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 189–196.

Cook, P. & Stevenson, S., 2009. An Unsupervised Model for Text Message Normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*. CALC '09. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 71–78.

Gamallo, P., García, M. & Campos, J.R.P., 2013. A Method to Lexical Normalisation of Tweets. *Tweet-Norm@ SEPLN*.

Han, B. & Baldwin, T., 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pp. 368–378.

Han, B., Cook, P. & Baldwin, T., 2012. Automatically Constructing a Normalisation Dictionary for Microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. EMNLP-CoNLL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 421–432.

Kobus, C., Yvon, F. & Damnati, G., 2008. Transcrire les SMS comme on reconnaît la parole. In *Actes de la Conférence sur le Traitement Automatique des Langues (TALN'08)*. pp. 128–138.

Krawczyk, S. & Raghunathan, K., 2009. Investigating sms text normalization using statistical machine translation.

Li, C. & Liu, Y., 2012. Normalization of text messages using character-and phone-based machine translation approaches. In *Thirteenth Annual Conference of the International Speech Communication Association*. Available at: http://www.isca-speech.org/archive/interspeech_2012/i12_2330.html.

Liu, F. et al., 2011. Insertion, Deletion, or Substitution?: Normalizing Text Messages Without Pre-categorization nor Supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 71–76.

Liu, F., Weng, F. & Jiang, X., 2012. A Broad-coverage Normalization System for Social Media Language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. ACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 1035–1044.

Pennell, D. & Liu, Y., 2011. A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. pp. 974–982.

Pennell, D.L. & Liu, Y., 2010. Normalization of text messages for text-to-speech. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. pp. 4842–4845.

Porta, J. & Sancho, J.-L., 2013. Word Normalization in Twitter Using Finite-state Transducers. *Tweet-Norm@ SEPLN*, 1086, pp.49–53.

Ruiz, P. & Cuadros, M., 2013. Lexical normalization of spanish tweets with preprocessing rules, domain-specific edit distances, and language models. *Proceedings of the Tweet*. Available at: https://hal.archives-ouvertes.fr/hal-01099250/.

Ruiz, P. & Cuadros, M., 2014. Lexical Normalization of Spanish Tweets with Rule-Based Components and Language Models. *del Lenguaje Natural*. Available at: https://hal.archives-ouvertes.fr/hal-01099241/.

Whitelaw, C. et al., 2009. Using the Web for Language Independent Spellchecking and Autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*. EMNLP '09. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 890–899.

Wong, W., Liu, W. & Bennamoun, M., 2006. Integrated Scoring for Spelling Error Correction, Abbreviation Expansion and Case Restoration in Dirty Text. In *Proceedings of the Fifth Australasian Conference on Data Mining and Analystics - Volume 61*. AusDM '06. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., pp. 83–89.