Universidad de la República
Facultad de Ingeniería
Instituto de Computación

Extracción de eventos en una ciudad a partir de redes sociales

Proyecto de Grado

Raúl Speroni - Martín Steglich
Tutor: Juan José Prada

Uruguay, Marzo de 2017.

Resumen

Las grandes ciudades y áreas metropolitanas son sistemas complejos con conexiones entre sus ambientes e individuos. Frente a la creciente urbanización, los servicios públicos requieren de mayor planificación y mecanismos de decisión que a su vez faciliten la participación ciudadana. Cotidianamente en Internet los ciudadanos se expresan sobre eventos relacionados a la ciudad. Esta información tiene gran valor por su inmediatez, diversidad de puntos de vista e impacto en la opinión pública.

Las tecnologías de la información permiten imaginar otro tipo de interfaces para la comunicación entre ciudadanos e instituciones. Interfaces capaces de extraer información útil aún si esta información no fue dirigida a las instituciones correspondientes.

En este trabajo se construye un sistema paramétrico y modular que combina diferentes técnicas para la extracción de eventos en una ciudad a partir de redes sociales. Usando la ciudad de Montevideo como caso de estudio fue posible identificar correctamente el 94 % de los Eventos relacionados a la limpieza con solo 4 % de falsos positivos.

Los avances tecnológicos sumados a datos abiertos de calidad permiten contruir este tipo de mecanismos de sensado de redes sociales que pueden ser relevantes para la gestión de una ciudad inteligente.

Palabras clave: ciudades inteligentes, *participatory sensing*, extracción de eventos, redes sociales, twitter, procesamiento del lenguaje natural, aprendizaje automático

Índice general

Capítulo 1

Introducción

1.1. Motivación

Actualmente estamos frente a dos fenómenos sociales relevantes para la historia de la humanidad: la aceleración de la urbanización y la revolución digital. Según la ONU [44] más de la mitad de la población del planeta vive en ciudades; y en Uruguay, el porcentaje es del 95 %. El acceso a Internet desde dispositivos móviles y su uso para consumir y difundir información por parte de los ciudadanos ha crecido explosivamente en Uruguay [51] y el mundo [21].

Conforme crece la población crecen los desafíos. Cada vez más, las grandes ciudades y áreas metropolitanas son vistas como sistemas complejos con conexiones entre sus ambientes e individuos. Los servicios como el tráfico, el transporte público, la recolección de residuos y la seguridad pública, entre otros, requieren de mayor planificación y mecanismos de decisión dinámicos que tomen en cuenta la inclusión de procesos de participación ciudadana [8].

Cotidianamente se expresa la perspectiva de los ciudadanos sobre Eventos relacionados a la ciudad en blogs y redes sociales. La información volcada por los ciudadanos, cada vez más frecuente con la utilización de dispositivos móviles, no siempre recorre los canales formales dispuestos por las diferentes organizaciones y sin embargo tiene gran valor de sensado por su inmediatez, diversidad de puntos de vista e impacto en la opinión pública.

En los últimos años se han visto en varios países del mundo, y también en Uruguay [15], iniciativas de transformación de "ciudades tradicionales" en ciudades inteligentes. Según Bouskela en [8] una ciudad inteligente se caracteriza entre otros aspectos por:

- Optimizar la asignación de recursos.
- Generar procedimientos eficientes.
- Suministrar información necesaria y transparente.
- Permitir mayor participación de la sociedad civil.
- Tener un alto grado de satisfacción entre sus habitantes.

El tránsito hacia las ciudades inteligentes incluye en general: sensores automáticos dispersos geográficamente que automatizan algunas tareas, interfaces web o aplicaciones móviles que permiten al ciudadano interactuar con la institución y el uso de redes sociales como una especie de mesa de informes global, donde, mediante un ardua gestión, se direccionan inquietudes hacia los diversos mecanismos establecidos.

El estado del arte en disciplinas de las tecnologías de la información como el Procesamiento del Lenguaje Natural, permiten imaginar otro tipo de interfaces para la comunicación entre ciudadanos e instituciones. Interfaces capaces de extraer información útil para la gestión de la ciudad aún si esta información no fue generada con la intención de ser trasmitida formalmente o no fue dirigida a la institución correspondiente. Es posible imaginar, y es objeto de este trabajo construir mecanismos de sensado de redes sociales donde una simple queja o comentario en Internet pueda convertirse en un indicador relevante para la gestión de una ciudad.

1.2. Objetivo

El objetivo de este trabajo es entonces desarrollar un sistema de extracción de información a partir de texto publicado en redes sociales. El sistema será parametrizable y deberá instanciarse para obtener Eventos que ocurren en una ciudad y que hacen referencia a un tema dado.

1.3. El problema

El problema consiste en el desarrollo de un sistema que permita extraer y procesar automáticamente información generada por los ciudadanos de una determinada ciudad en redes sociales. La información de interés es la vinculada a Eventos de un cierto dominio o temática.

A los efectos de este trabajo podemos definir Evento como una actividad del mundo real que ocurre durante cierto período de tiempo en cierto espacio geográfico. Los Eventos que se buscarán identificar serán los relacionados exclusivamente a un dominio, como por ejemplo transporte, recolección de residuos u otros servicios de una ciudad especifica.

1.4. Plan de trabajo

Al estar el equipo de trabajo conformado únicamente por dos personas se utilizó una metodología de trabajo ágil, con una reunión semanal presencial durante los primeros meses de trabajo. De forma progresiva se fueron aumentando la cantidad de reuniones semanales hasta llegar a cuatro reuniones presenciales semanales durante los meses de más trabajo.

A su vez, durante toda la ejecución del proyecto las reuniones entre el tutor y el equipo de trabajo fueron quincenales.

La ejecución de este proyecto tuvo una duración aproximada de diez meses, dividiéndose el trabajo en ocho tareas principales:

1. Análisis de las distintas fuentes de datos y normalización de la información: se analizaron las posibles fuentes de datos disponibles (Twitter, Facebook, blogs) realizando una normalización de los datos.

2. Prueba de las diferentes API (Sección 5.1) provistas por Twitter y ajuste de criterios de búsqueda para los tweets: los criterios de búsqueda ajustados durante esta tarea fueron los criterios utilizados durante todo el proyecto. Este ajuste se realizó mediante la utilización de las API provistas por Twitter.

3. Recolección de tweets con los criterios de búsqueda definidos, utilizando la Streaming API (5.1.8) de Twitter: una vez definidos los criterios de búsqueda se comenzó a construir el *corpus* mediante la importación de tweets. Este *corpus* fue el utilizado durante la implementación y obtención de resultados.

4. Estudio de trabajos relacionados con la extracción de eventos. Construcción del estado del arte (Capítulo 3): Se estudiaron diferentes trabajos cuyo campo de conocimiento está relacionado con este trabajo.

5. Estudio de recursos de PLN (Sección 2.2) disponibles: se investigaron diferentes herramientas de PLN que fueron utilizadas durante la construcción del sistema.

6. Diseño y desarrollo: se diseñó y desarrolló el sistema para este proyecto.

7. Validación de resultados sobre un *corpus* de pruebas: a medida que se fueron desarrollando los módulos que componen el sistema se comenzó con la ejecución de pruebas y obtención de resultados sobre un *corpus* de evaluación.

8. Elaboración del informe: durante toda la ejecución del proyecto se llevó a cabo la elaboración del informe. A medida que fueron completándose las diferentes etapas del proyecto, se fueron generando documentos intermedios validados por el tutor.

Estas ocho tareas que formaron parte del proceso se distribuyeron a lo largo del tiempo como se muestra en la Tabla 1.1.

Tarea/Mes	**Abr**	**May**	**Jun**	**Jul**	**Ago**	**Set**	**Oct**	**Nov**	**Dic**	**Ene**
1										
2										
3										
4										
5										
6										
7										
8										

Tabla 1.1: Dieagrama de Gantt seguido durante el proyecto

1.5. Organización del informe

El presente informe se estructura de la siguiente forma: el Capítulo 2 plantea los conceptos básicos utilizados durante el trabajo.

En el Capítulo 3 se presenta el estado del arte en los campos de conocimiento relacionados a este trabajo.

En el Capítulo 4 se presenta en profundidad el problema planteado para el proyecto y se explica su alcance.

En el Capítulo 5 se describe las principales herramientas tecnológicas utilizadas, se da una descripción general de la arquitectura de la solución planteada junto con una descripción en detalle de la construcción de cada una de las partes que componen el sistema.

En el Capítulo 6 se exponen los resultados de los diferentes módulos que componen el sistema junto con los resultados del sistema en general.

Por último, el Capítulo 7 plantea las conclusiones y propuesta de trabajo futuro y extensiones.

Capítulo 2

Marco teórico

El presente capítulo plantea algunos conceptos básicos utilizados durante este trabajo. De esta forma se da una introducción a los temas tratados en el informe para su correcto entendimiento. Primero se definen algunos conceptos sobre el Aprendizaje Automático. Luego los conceptos básicos del Procesamiento del Lenguaje Natural (PLN), pasando por las diferentes tareas del área utilizadas en el presente proyecto de grado. Finalmente se detallan las ideas básicas detrás de las Redes Sociales.

Tanto el Aprendizaje Automático como el PLN son subdisciplinas de la Inteligencia Artificial (IA). A su vez, como se explica en este capítulo, el Aprendizaje Automático tiene su utilización dentro del PLN.

2.1. Aprendizaje Automático

La principal idea del Aprendizaje Automático es hacer aprender a una computadora en base a la experiencia (datos).

Las tareas del Aprendizaje Automático más utilizadas son:

- Clasificación: consiste en determinar a qué categoría de un conjunto de categorías pertenece un objeto dado. En 2.2.4 se explica más sobre la clasificación, más enfocado al PLN.
- Regresión: es un método estadístico para predecir la probabilidad de un ejemplo de pertenecer a cierta categoría de un conjunto de categorías.

Generalmente el Aprendizaje Automático se divide en dos etapas. La primera de ellas es el entrenamiento, en donde se buscan patrones en los ejemplos disponibles y se intenta inferir un concepto general. La segunda etapa corresponde a la evaluación donde se mide el rendimiento del modelo elaborado en la etapa de entrenamiento.

El Aprendizaje Automático se puede dividir en diferentes tipos de técnicas, siendo las principales:

- Aprendizaje no supervisado: implica aprender de un conjunto de datos que no tienen etiquetas, y por lo tanto se basa más en encontrar patrones que en predicción.
- Aprendizaje supervisado: los datos contienen la información que se está modelando, esta es el objetivo que se desea predecir. Este tipo de aprendizaje es el que se utiliza durante este trabajo, ya que los datos poseen la información que se quiere predecir y esta es utilizada para entrenar el modelo.

- Aprendizaje semisupervisado: este tipo de aprendizaje combina ambos enfoques.

2.2. Procesamiento de Lenguaje Natural

En [18] se define el PLN como una subdisciplina de la Inteligencia Artificial, compuesta por un conjunto de métodos y técnicas eficientes desde un punto de vista computacional para la comprensión y generación del lenguaje natural.

El PLN suele dividirse en diferentes etapas, siendo las más clásicas:

Fonética y fonología

Es el estudio de los sonidos lingüísticos, cómo las palabras son pronunciadas en términos de secuencias de sonidos.

Análisis morfológico

Es el estudio individual de la estructura de las palabras en el que se construye una representación de cada una.

En la Figura 2.1 se puede observar un ejemplo de análisis morfológico para la palabra "Gatitos". Este análisis refleja que la palabra "Gatitos" es un derivado de la palabra "gato", que su género es masculino (etiqueta "Masc"), que se encuentra en plural ("Pl") y que es un diminutivo ("Dim").

Gatitos -> gato+Masc+Pl+Dim

Figura 2.1: Ejemplo de análisis morfológico para la palabra Gatitos, obtenido de [19]

Análisis léxico

El análisis léxico tiene como objetivo atribuir su categoría gramatical a cada palabra de una oración.

En la Figura 2.2 se muestra un ejemplo de análisis léxico para la oración "El gato come pescado.".

```
El/DET gato/NOM come/VB pescado/NOM ./SP
```

Figura 2.2: Ejemplo de análisis léxico, obtenido de [16]

Análisis sintáctico

El análisis sintáctico tiene como objetivo la construcción de un árbol que muestra una representación estructurada de cómo se relacionan las palabras de una oración.

Para este análisis se tienen dos componentes:

- Una representación declarativa del lenguaje, llamada gramática.
- Un proceso, llamado parser, que compara la gramática y las oraciones para producir el árbol.

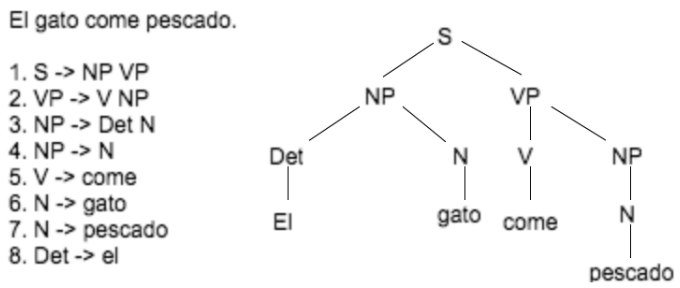


Figura 2.3: Ejemplo de análisis sintáctico

En la Figura 2.3 se observa un ejemplo de análisis sintáctico donde se puede ver la grámatica definida para el lenguaje, la oración a analizar y el árbol sintáctico resultado.

Análisis semántico

En el análisis semántico se le agrega significado a las estructuras formadas por el análisis sintáctico.

Discurso

Es el estudio de las unidades mayores a la oración. El significado de una oración puede depender de las oraciones que la preceden y también puede influir en el significado de la sentencia que le sigue.

El PLN tiene varias aplicaciones y usos, de los cuales algunos tienen vinculación directa en la vida real. En las siguientes secciones se explica aquellas que son particularmente relevantes para este trabajo.

2.2.1. Recuperación de Información

Según Manning, Raghavan y Schütze en [12], el concepto de Recuperación de Información (IR, por sus siglas en inglés) puede ser muy amplio, ya que quitar una tarjeta de crédito de la billetera para poder anotar su número se puede considerar como IR.

Sin embargo, como un caso de estudio académico, en [12] lo definen como: IR es encontrar material (usualmente documentos) de una naturaleza no estructurada (usualmente texto) a partir de grandes colecciones (usualmente almacenados en computadoras) que satisface una necesidad de información.

La frase "de una naturaleza no estructurada" refiere a datos que no son semánticamente claros y no siguen una estructura que permita ser interpretada fácilmente por una computadora. Este tipo de datos son lo opuesto a los datos estructurados, los que tienen como un ejemplo claro a las bases de datos relacionales.

Según [20] la IR consiste, básicamente, en tres pasos:

- Elegir un conjunto de palabras (consulta).
- Buscar las palabras escogidas en cada documento de una colección de documentos.
- Devolver los documentos que cumplan con la condición.

2.2.2. Extracción de Información

Como mencionan Piskorski y Yangarber en [34], las últimas décadas han sido testigo del aumento en la cantidad de información textual en formato digital almacenada en Internet. Una gran parte de esa información, proveniente por ejemplo de diarios digitales o redes sociales, es transmitida de forma no estructurada, lo que dificulta las búsquedas en esos medios. Esto dio lugar a una necesidad creciente de técnicas de análisis y descubrimiento de información relevante de textos no estructurados, conduciendo así a la aparición de técnicas de Extracción de Información (IE, por sus siglas en inglés).

La tarea de IE consiste en identificar, ignorando información no relevante, un conjunto predefinido de conceptos en un dominio específico, siendo este dominio un *corpus* de textos junto con una necesidad de información claramente especificada. En otras plabras, IE trata de extraer información estructurada a partir de texto no estructurado.

En la Figura 2.4, obtenida de [34], se puede observar un ejemplo de IE en Eventos violentos a partir de noticias, donde el interés está en identificar los principales actores del evento, la ubicación y la cantidad de personas afectadas.

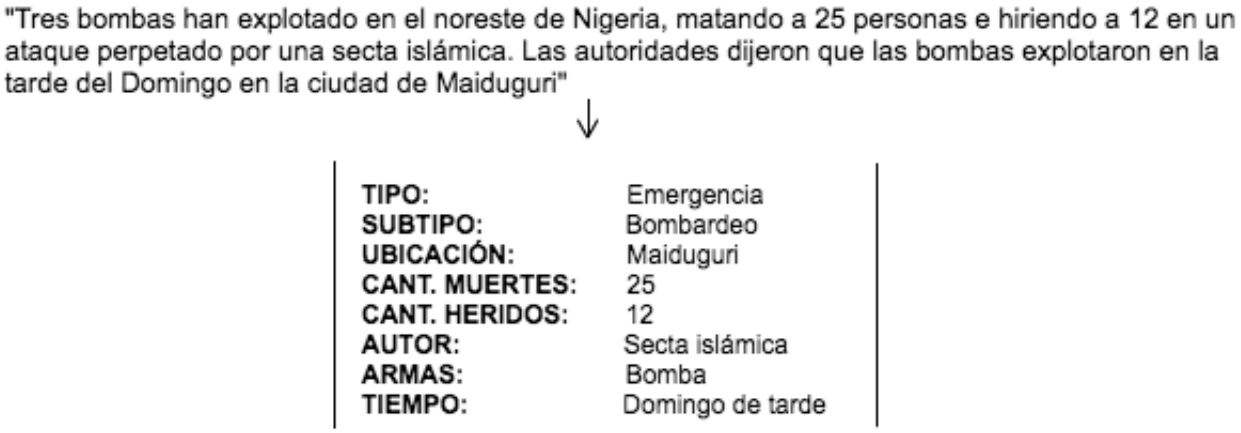


Figura 2.4: Ejemplo de extracción automática de información obtenido de [34]

Por lo general, los sistemas de IE se complementan con el IR. De forma tal que, utilizando primero IR se obtienen documentos relevantes para un problema y luego con IE se extrae la información relevante de estos documentos.

2.2.3. Reconocimiento de Entidades con Nombre

Reconocimiento de Entidades con Nombre (NER) es una tarea importante de los sistemas de extracción de información, que consiste en la correcta identificación de nombres en textos para clasificarlos dentro de categorías de interés predefinidas. Dentro de estas categorías las más comunes son: personas, organizaciones y ubicaciones. También es usual encontrar categorías referentes a expresiones temporales (fechas, horas) y a cantidades (monedas, medidas, porcentajes, números) [27].

Por lo general, las aplicaciones que realizan NER toman como entrada una porción de texto como la que se muestra en el Ejemplo 2.2.1:

Ejemplo 2.2.1 *La Facultad de Ingeniería, que se encuentra en el Parque Rodó, tiene como decana a María Simón.*

Produciendo una porción de texto anotado como en el Ejemplo 2.2.2 donde se destacan las entidades encontradas:

Ejemplo 2.2.2 *La **[ORG Facultad de Ingeniería]**, que se encuentra en el **[LOC Parque Rodó]**, tiene como decana a **[PER María Simón]**.*

Para la oración del ejemplo precedente se identificaron tres entidades: Facultad de Ingeniería que representa una organización, Parque Rodó representa una ubicación y María Simón una persona.

2.2.4. Clasificación

La clasificación consiste en determinar a qué clase pertenece un objeto dado. En este sentido, muchas tareas del PLN pueden verse como problemas de clasificación, como por ejemplo determinar el idioma de un documento, determinar si una opinión es positiva o negativa, etc.

Dentro de los clasificadores hay dos grandes líneas:

- Reglas.
- Aprendizaje automático.

Los clasificadores basados en reglas buscan construir reglas que capturen el conocimiento de expertos para resolver una problemática particular. Las reglas construídas intentan generalizar lo suficiente pero sin dejar de lado los casos particulares.

Los clasificadores basados en aprendizaje automático toman como entrada un conjunto de datos (*corpus*) y devuelven la función de clasificación. El *corpus* suele dividirse en dos conjuntos, entrenamiento y evaluación. Usualmente el primero tiene un tamaño mayor que el segundo, es común tomar el 80 % del total para el *corpus* de entrenamiento y un 20 % para el de evaluación.

El *corpus* de entrenamiento se utiliza para generar un modelo (función de clasificación) y luego se utiliza este modelo para calcular la categoría[1] de cada uno de los elementos del *corpus* de evaluación. Finalmente se evalúa la performance sobre el mismo *corpus* utilizando las siguientes medidas, tomando en cuenta una de las clases a clasificar:

- *Precision*: se puede considerar como una medida de la exactitud de los clasificadores. Se calcula como el cociente entre la cantidad de instancias que se clasificaron correctamente y la cantidad de instancias que se clasificaron como pertenecientes a la clase:

$$\mathbf{VP\ /\ (VP\ +\ FP)}\ [VP = Verdaderos\ Positivos,\ FP = Falsos\ Positivos]$$

- *Recall*: se puede considerar como una medida de la completitud de los clasificadores. Se calcula como el cociente entre la cantidad de instancias que se clasificaron correctamente y la cantidad de instancias que pertenecen a la clase:

$$\mathbf{VP\ /\ (VP\ +\ FN)}\ [FN = Falsos\ Negativos]$$

[1] clase a la que pertenecen

- *Fall-out*: se puede ver como la probabilidad de clasificar positivamente instancias negativas. Se calcula como el cociente entre la cantidad de instancias que se clasificaron incorrectamente y la cantidad de instancias que no pertenecen a la clase:

$$\textbf{FP / VN} \quad [VN = Verdaderos\ Negativos]$$

- *F1-score*: es la medida armónica entre *precision* y *recall* . Se calcula como:

2*Recall*Precison / (Recall + Precision)

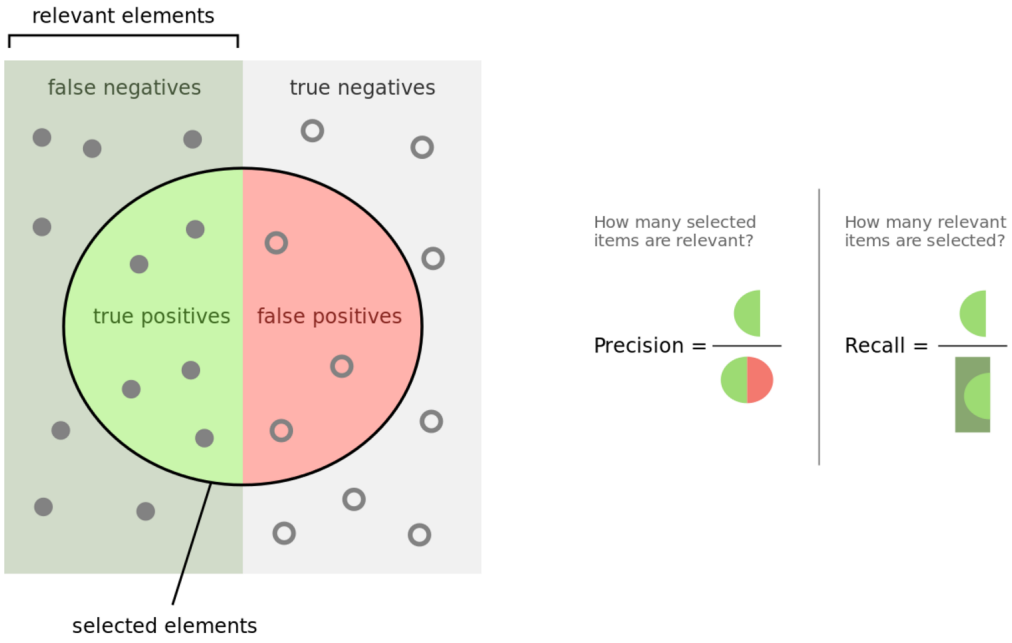


Figura 2.5: Representación gráfica de *precision* y *recall*. Obtenida de [66]

Estas medidas son utilizadas, también, para evaluar los resultados de un sistema de IR. La Figura 2.5, muestra una representación gráfica de cómo se calculan *precision* y *recall*, para un sistema de IR.

A su vez, el modelo puede contener diferentes parámetros que son calculados experimentalmente utilizando el *corpus* de entrenamiento. Para ello existen dos enfoques diferentes, según [17]:

- Held-out: se separa una parte del *corpus* de entrenamiento y se lo utiliza para evaluar.

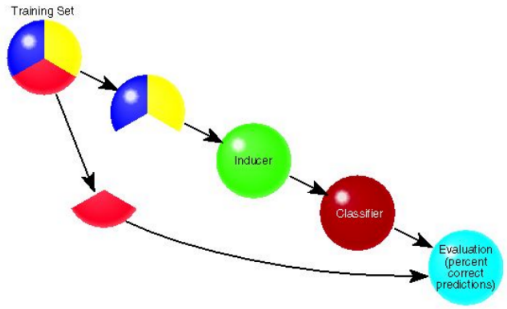


Figura 2.6: Representación gráfica de Held-out, obtenida de [17]

- *Cross validation*: se divide el *corpus* de entrenamiento en K partes iguales. Se entrena sobre (k-1) partes y se evalúa sobre la parte restante. Se repite el proceso para todas las partes y se calcula la media.

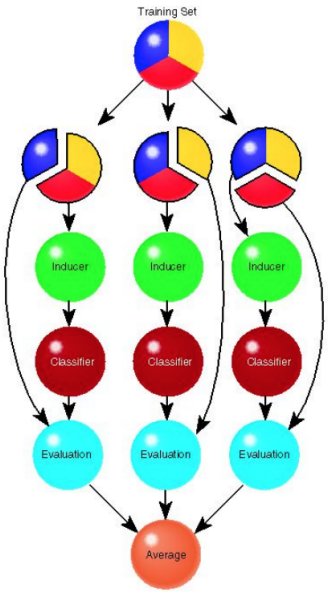


Figura 2.7: Representación gráfica de *cross validation*, obtenida de [17]

Existen diferentes técnicas de clasificación, algunas de estas son explicadas en 2.2.4.1 y 2.2.4.2, a modo de introducción a las técnicas utilizadas en este trabajo.

2.2.4.1. Árbol de decisión

En [55] se presentan a los árboles de decisión como un método no paramétrico de aprendizaje supervisado utilizado tanto para clasificación como para regresión. El objetivo es crear un modelo que prediga el valor de una instancia mediante reglas simples inferidas a partir de las características de la información.

Como se muestra en la Figura 2.8, los árboles de decisión aprenden de los datos para aproximar una curva sinusoidal con un conjunto de reglas *if-then-else*. Cuanto más profundo es el árbol, más complejas y más ajustadas a los datos son las reglas de decisión.

Los árboles de decisión clasifican instancias ordenándolas en un árbol desde la raíz hasta alguna hoja que determina la clase de la instancia. Cada nodo en el árbol evalúa algún atributo (*feature*) de la instancia.

Es de interés para los árboles de decisión determinar la importancia de cada atributo en el contexto de la clasificación. Según Tom M. Mitchell en [42] esta importancia se define como ganancia de la información o *gain* en términos de la entropía. La entropía en este contexto caracteriza la impureza de un conjunto de instancias. Dado un conjunto S con instancias positivas y negativas respecto a alguna clase objetivo, la entropía de S se define como:

$$Entropia(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \tag{2.1}$$

La ganancia de información ($Gain$) se puede definir entonces como la reducción de la entropía causada por el particionado de instancias según el atributo que se está evaluando:

$$Gain(S, A) \equiv Entropia(S) \; - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} Entropia(S_v) \tag{2.2}$$

donde $Valores(A)$ es el conjunto de todos los posibles valores para el atributo A y S_v es el subconjunto de S para el que el atributo A tiene valor v.

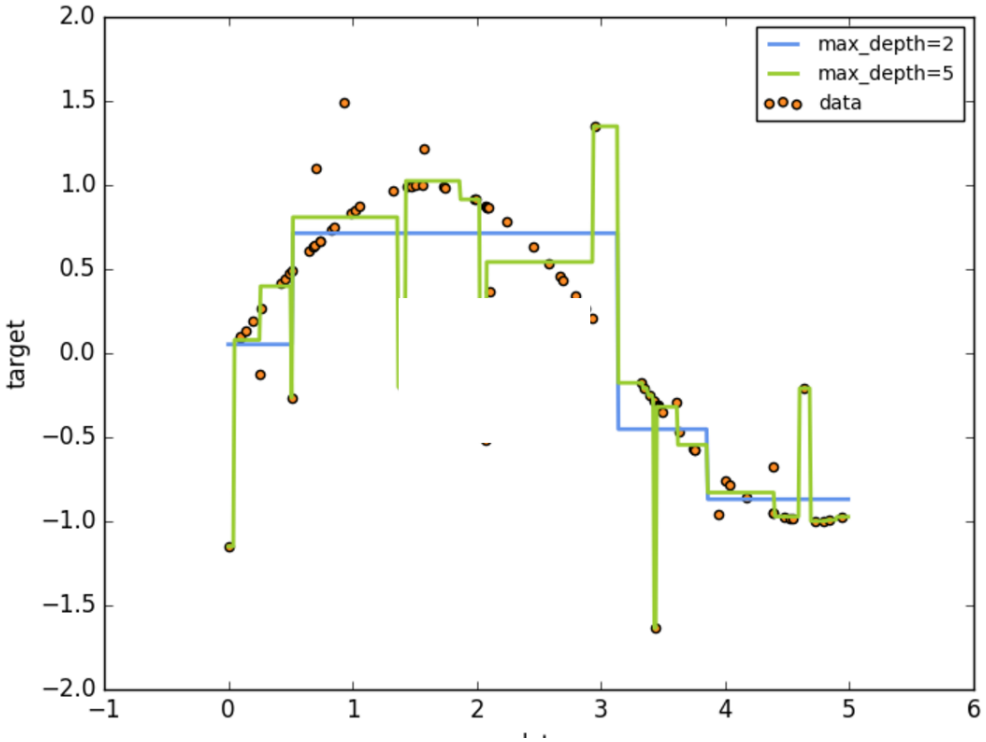


Figura 2.8: Ejemplo de curva obtenida por un árbol de decisión

2.2.4.2. Support Vector Machines

En [56] se presenta a Support Vector Machines (SVM) como una técnica de aprendizaje supervisdo del campo de aprendizaje automático, aplicable tanto para clasificación como para regresión. Esta técnica busca definir un hiperplano de separación entre los ejemplos.

Como se menciona en [59], la solución obtenida se basa únicamente en los puntos que se encuentran en el margen (distancia mínima entre el hiperplano de separación y los puntos más cercanos), los que son denominados "vectores de soportes", dándole así el nombre a la técnica.

Cuanto mayor sea el margen, menor es el error de generalización del clasificador y por lo tanto el hiperplano definido por la técnica es óptimo cuando el margen es máximo.

La Figura 2.9, obtenida de [56], muestra cómo funciona SVM, siendo el hiperplano resultado una recta y obteniendo una total separación de los datos de ejemplo, cosa que no siempre es posible ya que pueden quedar ejemplos mal clasificados.

2.2.5. Otros conceptos básicos del PLN

En esta sección se da una introducción a algunos conceptos básicos del PLN utilizados durante el trabajo y que no fueron explicados anteriormente.

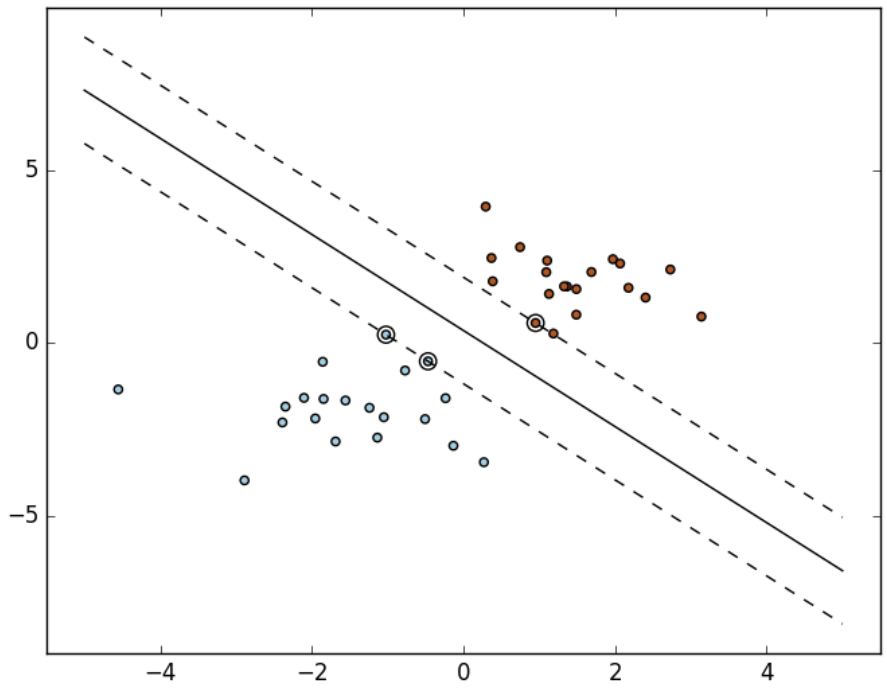


Figura 2.9: Ejemplo de hiperplano de separación de SVM. Los puntos resaltados son los llamados "vectores de soporte"

2.2.5.1. Separación en *Tokens*

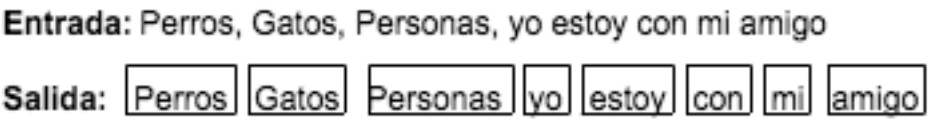


Figura 2.10: Ejemplo de separación en tokens

La separación en *tokens* es el proceso de dividir un texto dado en unidades que representan, por lo general, palabras o números. Al mismo tiempo, es común que también se eliminen los símbolos de puntuación.

En la Figura 2.10 se muestra un ejemplo de entrada y salida para una separación en *tokens*.

2.2.5.2. Etiquetado gramatical de palabras

En [62] se define POS (Part-Of-Speech) Tagger como una pieza de software que procesa texto en algún lenguaje y asigna etiquetas Part-Of-Speech[2] para cada una de las palabras.

En el Ejemplo 2.2.3 se muestra un ejemplo de entrada a un POS Tagger, obteniendo como resultado lo mostrado en el Ejemplo 2.2.4

Ejemplo 2.2.3 *El gato come pescado.*

[2]nombre, adjetivo, verbo, etc

Ejemplo 2.2.4 $El_{\boldsymbol{Det}}\ gato_{\boldsymbol{Nom}}\ come_{\boldsymbol{Verb}}\ pescado_{\boldsymbol{Nom}}$.

2.2.5.3. Stopwords

otro	alguno
un	sos
el	vaya
avenida	tienen
quizá	cierto
camino	hace
tan	aquí

Tabla 2.1: *Stopwords*

Las *stopwords* son palabras que no agregan ninguna información al texto en cuestión, por lo que generalmente son eliminadas antes de comenzar. Las *stopwords* del ejemplo de la Figura 2.10 son: yo, con y mi.

En Tabla 2.1 se muestran algunas de las *stopwords* utilizadas para este trabajo.

2.2.5.4. Stemming o lematización

Formas	**Lema**
jugaba, jugando, jugará	jugar
alto, alta, altos	alto
mesa, mesas	mesa

Tabla 2.2: Ejemplo de lematización

Según [63] el objetivo de la lematización es reducir las formas inflexionadas y las formas derivadas de una palabra a una forma básica común, como se muestra en la Tabla 2.2

2.2.5.5. N-gramas

Dada una secuencia, un N-grama es una subsecuencia de N elementos. Se utilizan en muchas áreas del conocimiento. Por ejemplo en PLN, se puede utilizar para predecir la siguiente palabra de una oración a partir de las $N-1$ anteriores.

En la Figura 2.11 se muestra un ejemplo de N-gramas cuando $N = 1$ (unigramas), $N = 2$ (bigramas) y $N = 3$ (trigramas).

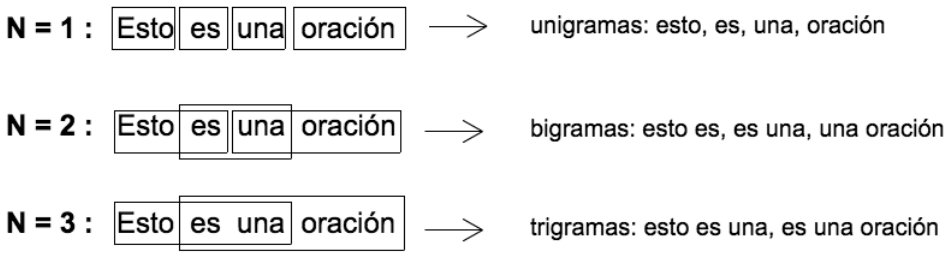


Figura 2.11: Ejemplo de unigramas, bigramas y trigramas, obtenido de [65]

2.3. Red Social

Según [1], las redes sociales (RRSS) implican el uso de Internet para conectar a los usuarios con sus amigos, familiares y conocidos de la vida real. Las RRSS, si bien no tienen como principal objetivo conocer gente nueva, son utilizadas para ello.

Generalmente, cada uno de los "amigos" (*Facebook*) o "seguidores" (*Twitter*) se conectan entre sí. Como en la vida real, las conexiones entre personas no son únicamente de a dos, sino que forman una red de conexiones. Estos sitios son útiles para difundir información, fotos y videos y estar en contacto con personas con las que no se interactúa todo el tiempo.

Las RRSS más conocidas son *Facebook*[3], *Twitter*[4], *Instagram*[5] y *LinkedIn*[6]. Todas ellas proveen diferentes mecanismos para la obtención de información que puede resultar muy útil para el propósito de este trabajo.

En el Capítulo 3 se explica un poco más sobre RRSS.

[3] www.facebook.com
[4] www.twitter.com
[5] www.instagram.com
[6] www.linkedin.com

Capítulo 3

Estado del Arte

En este capítulo se describe el estado del arte en trabajos para la extracción de Eventos según se definen en la Sección 1.3. Se analiza el caso de texto escrito en general y, para las redes sociales en particular además de su aplicación en ciudades.

3.1. Extracción de Eventos a partir de texto escrito

Existe abundante investigación previa sobre extracción de Eventos a partir de texto escrito. Estos trabajos se pueden categorizar según tipos de eventos, fuentes de datos y métodos utilizados [4].

En las siguientes secciones se analizan diferentes técnicas categorizadas por tipo de fuente de datos. En particular se exponen técnicas clásicas para medios escritos tradicionales y técnicas novedosas para medios electrónicos masivos como son las redes sociales.

3.1.1. Medios tradicionales

Previo a la popularización de las redes sociales, la fuente de datos principal para la investigación de extracción de Eventos fueron los medios tradicionales escritos, como por ejemplo periódicos o libros. La mayoría de las técnicas utilizadas para estas fuentes de datos tienen en común el uso de un *corpus* de documentos existente, cuyo contenido si bien no necesariamente es estructurado, tiene ciertas características lingüísticas que lo simplifican, como ser un bajo número de errores y pocas o nulas abreviaciones no estandarizadas.

A continuación se presentan algunos ejemplos divididos en dos tipos de técnicas que se diferencian principalmente en cómo son representados los documentos.

3.1.1.1. Técnicas *Document-Pivot*

Esta familia de técnicas agrupan documentos en base a su similitud textual.

Tradicionalmente la representación de documentos involucra las técnicas de *term vectors*, que representa un documento como vector de términos, y *bag of words*, donde un documento es representado como un conjunto de términos.

En la representación como *term vectors* cada dimensión del vector representa un término y tiene valor positivo si ocurre en el documento.

La representación como *bag of words* consiste en un conjunto de palabras que aparecen en el documento.

Típicamente, a cada palabra se le asigna un peso usando Tf-Idf, una medida numérica que expresa cuán relevante es un término para un documento dentro del *corpus* [53]. Este tipo de representaciones descarta el orden de las palabras, así como características sintácticas y semánticas.

Como alternativa, la representación como *named entity vectors* intenta extraer información contestando las preguntas quién, cómo, cuándo y dónde [37]. Existen también integraciones de ambas técnicas utilizando *mixed vectors* que son representaciones que mezclan varios enfoques [67].

En los trabajos [68] y [2] se comenta que se puede dividir el problema de extracción de Eventos en: Detección retrospectiva de Eventos (RED) y Detección de nuevos Eventos (NED). RED se enfoca en encontrar Eventos sin identificar de un conjunto acumulado históricamente, mientras que NED implica el descubrimiento de Eventos a partir de un flujo de documentos casi en tiempo real.

La Detección retrospectiva de Eventos involucra algoritmos de *clustering* que usan todo el *corpus* para organizar los documentos por tema. Técnicas de *clustering* jerárquico (HCA) han sido ampliamente utilizadas para esta tarea [33].

Los algoritmos HCA parten de un conjunto por documento, estos conjuntos se van uniendo a partir de cierta función de distancia hasta que se satisface algún criterio de parada elegido.

Otros algoritmos como el k-means, que agrupa los diferentes documentos en K grupos utilizando la media como función de distancia, y sus variantes como k-median y k-median++, también han sido utilizados para resolver problemas RED [7].

Por otra parte, la Detección de nuevos Eventos no puede ser expresada como una consulta explícita ya que la información sobre el Evento es desconocida.

A diferencia de RED, NED debe tomar decisiones a medida que llegan nuevos documentos. Por lo tanto las técnicas usadas son típicamente basadas en *algoritmos greedy*, es decir, algoritmos que siguen heurísticas basadas en óptimos locales, que procesan el flujo de información secuencialmente uniendo documentos a los más similares o creando nuevos conjuntos (*clusters*) si la medida de distancia excede un cierto límite [2]. En la práctica, estas técnicas son costosas en cuanto a tiempo y cómputo por lo que se han desarrollado otros enfoques. Un ejemplo, es el uso de una ventana de tiempo para la comparación de documentos partiendo del supuesto de que la ocurrencia de documentos sobre un mismo Evento debería suceder en cierto marco temporal. Otros usos incluyen limitar el número de términos por documento y emplear computación paralela [68, 47, 40].

Las técnicas y enfoques mencionados parten en general de la premisa que todos los documentos tienen importancia relativa para Eventos nuevos o ya conocidos. Estas técnicas son además costosas, haciéndolas poco adecuadas para grandes cantidades de información con una proporción elevada de ruido, como son los flujos de información de las redes sociales [5, 10, 32, 38].

3.1.1.2. Técnicas *feature pivot*

Las técnicas *feature pivot* modelan un Evento dentro de un flujo de textos como una actividad en ráfagas con ciertas características (*features*) destacándose por su frecuencia a medida que el Evento emerge. En estas técnicas, un Evento es representado como un conjunto de palabras clave que muestran una ráfaga en el conteo de ocurrencias según explica Kleinberg en [35]. El supuesto principal en el que se basan estas técnicas es que ciertas palabras relacionadas mostrarán un incremento en su uso a medida que el Evento

ocurre. Kleinberg lo ejemplifica con un conjunto de correos electrónicos durante un período de tiempo.

La tarea de detección de tendencias sobre medios escritos generalmente busca identificar temas nuevos o temas que tienen creciente importancia dentro del *corpus* [36]. En la misma línea, se han investigado diferentes técnicas de detección de ráfagas de Eventos en medios escritos tradicionales [35, 25, 29, 30, 64, 28].

El autor también propone en [35] un autómata para modelar los tiempos de llegada de documentos en flujos. El objetivo es identificar ráfagas de alta intensidad durante períodos cortos. Los estados del autómata corresponden a las frecuencias de palabras individuales, mientras que las transiciones representan las ráfagas ante un cambio sustancial de frecuencias.

Fung y otros modelan en [25] la ocurrencia de palabras como una distribución binomial, identificando ráfagas de palabras de acuerdo a un límite basado en ciertas heurísticas y agrupándolas para detectar ráfagas de eventos.

La transformada discreta de Fourier se usó en [29] para aplicar análisis espectral con el objetivo de categorizar *features* para diferentes características de los Eventos (importantes o no, periódicos o aperiódicos, etc).

En [57], Snowsill y otros presentan un enfoque online para detectar Eventos en flujos de noticias basados en tests de significancia estadística sobre frecuencias de n-gramas dentro de un marco de tiempo.

La aplicación directa de estas técnicas sobre un gran volumen de información con ruido como la proveniente de redes sociales no parece factible especialmente dado que no todas las ráfagas son de interés.

3.1.2. Redes Sociales

El uso de redes sociales como fuente de datos para la extracción de Eventos está vinculado a las posibilidades de consulta de información que ofrece cada plataforma. La no existencia de una API pública para consultar los contenidos de Facebook hace que esta red prácticamente no se utilice en este tipo de estudios. La extracción de Eventos a partir de entradas de blogs se ha realizado elaborando un *corpus* utilizando el formato RSS junto a otros métodos de extracción de información como puede verse en el trabajo de Okamoto y Kikuchi en [46]. La API de búsqueda de Twitter[1], bien documentada y de uso gratuito, así como la creciente popularidad de la plataforma en casi todos los países del mundo, hacen que sea la elección más frecuente y es la que finalmente se utilizará en este trabajo.

3.1.2.1. Twitter

Twitter es una red social en la que sus usuarios pueden compartir mensajes breves llamados tweets. Un tweet puede ser como máximo de 140 carácteres y contener adjuntos como fotografías y videos.

Dependiendo de la información disponible y la naturaleza del problema, la detección de Eventos en Twitter puede ser clasificada en técnicas para Eventos conocidos y técnicas para Eventos desconocidos.

Eventos desconocidos

[1] https://dev.twitter.com/rest/public

Cuando no hay información previa sobre el Evento a detectar, las técnicas a utilizar deben basarse en patrones temporales del flujo de información de Twitter para detectar Eventos del mundo real. La aparición de nuevos Eventos de interés general reflejarán en Twitter ciertas características como por ejemplo un uso repentino de algunas palabras clave. Este tipo de características que ocurren juntas frecuentemente pueden ser agrupadas como tendencias [41]. En Twitter existen, además, Eventos endógenos o tendencias no relacionadas a Eventos [45]. Las técnicas para la detección en estos casos deberán entonces discriminar la información relevante usando algoritmos escalables y eficientes.

Un sistema de procesamiento de noticias basado en Twitter llamado TwitterStand es propuesto por Jagan Sankaranarayana y otros en [54]. En ese trabajo se emplea un clasificador Naive Bayes para determinar si un tweet corresponde a una noticia o a información irrelevante y luego se utiliza un algoritmo de *clustering* basado en *term vectors* usando similaridad Tf-Idf para agrupar las noticias.

Phuvipadawat y otros en [48] presentan un método para obtener, agrupar y ordenar noticias a partir de Twitter. Se agrupan los tweets similares para formar un histórico, la similaridad está basada en Tf-Idf con pesos incrementados en sustantivos, *hashtag* y nombres de usuario. Los sustantivos son detectados usando el Reconocedor de entidades con nombre de Stanford (NER)[2] entrenado sobre un *corpus* de noticias tradicionales. Se usa además una combinación de número de seguidores en Twitter como medida de fiabilidad, número de retweets como indicador de popularidad y la actualidad del tweet para asignarle importancia relativa a cada cluster.

Eventos conocidos

La detección de Eventos conocidos está basada en la existencia de información sobre el Evento que se está buscando, como puede ser una ubicación de interés, un tema o un marco temporal específico. Esta información puede ser utilizada adaptando técnicas tradicionales de *IR* e *IE* de acuerdo a las características propias de los tweets.

Identificar Eventos polémicos que fueran origen de discusiones públicas en Twitter es el objeto del *framework* desarrollado por Popescu y Pennacchiotti en [49]. El *framework* está basado en *snapshots* de Twitter. Los *snapshots* se definen como ternas compuestas de una entidad objetivo, un período de tiempo, y un conjunto de tweets sobre la entidad en ese período.

Se distinguen entre *snapshots* sobre Eventos y aquellos irrelevantes usando árboles de decisión supervisados entrenados sobre un *corpus* anotado manualmente [24].

Para ordenar los *snapshots* referentes a Eventos se usa un modelo de controversias basado en un *Regression Algorithm* aplicado a una gran cantidad de *features*.

En [50] Takeshi Sakaki y otros emplearon el mismo *framework* con ciertas modificaciones. La idea principal es basarse en la importancia y el número de entidades encontradas sobre Eventos y el número encontrado en el resto. Según los autores, la mayoría de los *snapshots* relacionados a Eventos tienen poca cantidad de entidades importantes mientras que los demás *snapshots* tienen mayor cantidad de entidades no importantes.

Un buen ejemplo de uso se ve en [52], donde Sakaki y otros utilizaron tweets para detectar tipos específicos de Eventos como ser terremotos y tifones. Formularon el problema de detección de Eventos como un problema de clasificación entrenando una SVM sobre datos manualmente etiquetados para separar tweets positivos (terremotos y tifones) de los negativos (otros Eventos o no eventos). El análisis de cierta cantidad de tweets durante un período de tiempo, para terremotos y tifones, mostró una distribución exponencial de

[2]http://nlp.stanford.edu/ner/

Eventos. Los parámetros para esta distribución fueron estimados de datos históricos y fueron usados para computar una cantidad de tiempo de espera confiable antes de lanzar una alarma.

Por otra parte, se presenta en [5] un sistema de información aumentada sobre Eventos planificados utilizando mensajes de Twitter, mediante reglas simples y estrategias de construcción de consultas. Para identificar un Evento a partir de tweets, el sistema comienza con consultas simples y precisas derivadas de la descripción del Evento y sus aspectos asociados (marco temporal, descripción, ubicación). Manualmente se etiquetan los resultados de cada estrategia de consulta; para mejorar el *recall* se usa Tf-Idf para identificar términos y frases descriptivas. Estos términos y frases, además de *hashtags* y enlaces URL, son utilizados recursivamente para la construcción de nuevas consultas.

Alqhtani y otros combinan en [3] la extracción de Eventos de Twitter con el uso de *Image Mining*, una técnica que utiliza conceptos de *Computer Vision* e *Image Processing* para que las imágenes adjuntas a un tweet sean consideradas en la clasificación.

3.2. Extracción de Eventos en ciudades

El concepto de ciudades inteligentes, si bien no tiene una única definición es asociada por Chourabi y otros en [11] al uso de la información disponible en una ciudad para la resolución de problemas derivados del crecimiento demográfico, infraestructura, etc. Esta información es generada a diario por diversos sensores, como pueden ser los GPS del transporte público, las cámaras de tránsito, y otros dispositivos. Surge en este contexto la idea de *participatory sensing*, la recolección de información local, participativa y en tiempo real a partir de los dispositivos móviles conectados a Internet de los ciudadanos. Ambos conceptos se han estudiado con aplicaciones en los campos de la planificación urbana, salud pública, tráfico y manejo de recursos naturales [9].

3.2.1. Trabajos previos en Montevideo

El interés de la Ciudad de Montevideo en convertirse en una ciudad inteligente [15] así como su política de datos abiertos [13] ha permitido el surgimiento de algunas aplicaciones principalmente relacionadas a la recepción de reportes y denuncias sobre Eventos de interés para los ciudadanos.

Por Mi Barrio[3] es una plataforma desarrollada por la organización de la sociedad civil DATA[4], para realizar reclamos sobre difrentes aspectos de la ciudad. Permite a los usuarios enviar una nueva denuncia georreferenciada o sumarse a una ya existente. Mediante el convenio 5742/13[5] entre la Intendencia de Montevideo (IM) y DATA los reclamos ingresan de forma automática, utilizando servicios abiertos provistos por la IM, al Sistema Único de Reclamos de la IM (SUR) [14] y, cuando existe una actualización sobre el estado del reclamo, el usuario de la plataforma es informado mediante notificaciones.

El Buzón Ciudadano[6] es otra de las interfaces disponibles para la realización de reclamos o sugerencias a la IM que también interactúa con SUR.

[3]http://pormibarrio.uy/
[4]http://www.datauy.org/
[5]http://www.montevideo.gub.uy/aplicacion/resoluciones
[6]http://www.montevideo.gub.uy/formularios/buzon_ciudadano

CitiViva[7] y Barrios Activos[8] son soluciones similares para el envío de denuncias que también incluyen características de redes sociales como son el apoyo de vecinos a una denuncia o iniciativa y la posibilidad de compartir los reportes. En ambos casos no se reporta en sus sitios web cómo se canalizan las denuncias por vías formales.

[7] http://www.citiviva.com/
[8] https://barriosactivos.com

Capítulo 4

Análisis del problema

En este capítulo se describe en mayor profundidad el problema planteado y se explica el alcance definido para este trabajo. Se discuten también algunos aspectos que deberán ser parte de la solución.

4.1. Descripción

El problema consiste en el desarrollo de un sistema que permita extraer información generada por los ciudadanos de una determinada ciudad, vinculada a Eventos de un cierto dominio, en medios escritos digitales.

Existe una gran variedad de medios escritos sobre los que se podría estudiar Eventos en una ciudad. Como se vió en el Capítulo 3, los medios escritos se pueden categorizar en medios tradicionales (libros, prensa) y en medios electrónicos (redes sociales, emails, blogs). Dado el carácter temporal de los Eventos y la necesidad de limitar el alcance de este trabajo, se decidió utilizar únicamente la red social Twitter como fuente de datos. Esta red es conocida por la inmediatez con que aspectos de la vida cotidiana de las ciudades son reflejados en Internet.

Para este trabajo se eligió la ciudad de Montevideo, que más allá de ser nuestra ciudad de residencia, se destaca por la cantidad y calidad de los datos abiertos y georreferenciados que ofrece.

La limpieza de la ciudad es el dominio elegido. Identificamos esta problemática como uno de los aspectos más sensibles para la población y por lo tanto un tema para el que pueden existir una importante cantidad de denuncias en redes sociales que servirán como *corpus* de datos para este trabajo.

Como se mencionó en 3.2.1 existen mecanismos centralizados y establecidos para la recepción de reclamos y denuncias por parte de la Intendencia de Montevideo (IM). No obstante, no siempre son usados por quienes tienen denuncias para hacer. En muchos casos, como se muestra en la Figura 4.1, estas personas prefieren la inmediatez que ofrece una red como Twitter para denunciar alguna problemática de la ciudad. El principal problema con esta realidad es que la IM puede ignorar que la denuncia fue realizada, o el denunciante puede no enterarse si su reclamo está siendo atendido. A su vez, por lo general el denunciante utiliza este medio como forma de expresar una inquietud y no para plantear una denuncia formal por lo que muchas veces no espera que el reclamo sea atendido.

El desafío planteado es la recuperación de la información sobre limpieza en la ciudad de Montevideo volcada de forma pública en Twitter.

Figura 4.1: Tweets sobre limpieza en Montevideo

Esta información puede o no estar dirigida a la institución responsable o a sus representantes.

El objetivo central de este trabajo es la detección de Eventos a partir de la información recuperada que de otra manera serían ignorados por los sistemas corporativos establecidos por la IM. Es de interés además de la Detección de nuevos Eventos (NED), el enriquecimiento de información respecto a Detección retrospectiva de Eventos (RED), permitiendo otro nivel de análisis de las problemáticas y el reconocimiento de ciertos patrones que pueden colaborar con su resolución.

4.2. Formalización

Se estima que se publican en el mundo, un promedio de 6000 tweets por segundo. De ese universo, la solución a este problema debe lograr identificar aquellos tweets en español, que sean reclamos o denuncias relativos a la limpieza y sean de Montevideo.

Es posible formalizar el problema utilizando teoría de conjuntos.

Sea A el universo de tweets. Sean $B, C, D \in A$ tweets que son reclamos en español, tweets de limpieza y tweets de Montevideo respectivamente. Entonces como se ve en la Figura 4.2 el espacio de búsqueda es

$$E = B \cap C \cap D \tag{4.1}$$

Donde E es el conjunto de tweets útiles.

4.3. Solución

En esta sección se busca describir una solución que resuelva el problema planteado en la Sección 4.1. Se enumeran además algunos de los desafíos visualizados al momento de

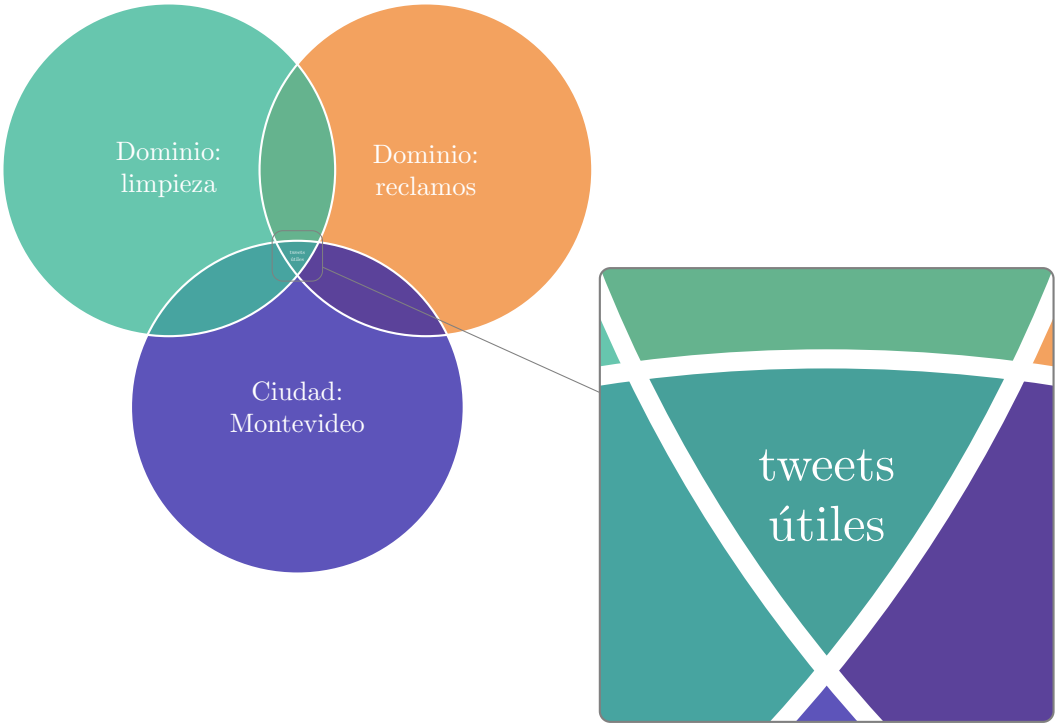


Figura 4.2: Formalización del problema

comenzar el trabajo.

4.3.1. Descripción

Una solución general al problema planteado debería poder instanciarse en una ciudad y en un dominio como los elegidos para este trabajo sin perder la posibilidad de ser usada en otros dominios o ciudades. Se pretende construir una plataforma que mediante configuración y parametrización pueda tener un campo amplio de aplicación, pudiendo extenderse por ejemplo a seguridad o tránsito.

A su vez, esta plataforma debiera poder extenderse para recibir información de diversas fuentes de datos (email, Facebook) y la salida producida, es decir los Eventos identificados, deberían poder integrarse fácilmente a otro tipo de sistemas, como por ejemplo, en el caso de Montevideo, al sistema SUR [14].

Se busca que la plataforma a construir agregue valor de forma automática a información que hoy las instituciones reciben por medio de redes sociales así como a la que no reciben directamente.

Interesa también que el sistema construido prevea mecanismos de visualización, análisis y gestión de los resultados. Se deberán presentar los datos de forma de maximizar su utilidad para los eventuales usuarios.

Es claro que la capacidad de cualquier sistema de recuperar Eventos sobre los distintos servicios de una ciudad a través de redes sociales está estrechamente vinculada a su gestión por parte de las instituciones responsables. La realidad será distinta si la población conoce cómo dirigir comentarios o denuncias en redes sociales a las instituciones o representantes.

Se buscará construir una solución de software que junto a ciertas políticas y acciones permita a una ciudad avanzar, al menos en parte, en la transformación hacia una ciudad inteligente.

4.3.2. Desafíos

Datos

Si bien es posible encontrar tweets relacionados a limpieza en Montevideo, estos no se encuentran en grandes cantidades por lo que puede resultar difícil la construcción de un *corpus* de datos que permita un correcto análisis de resultados.

Temática sensible

El dominio elegido, la limpieza, es un tema por demás sensible para la ciudadanía. La cantidad y calidad de la información disponible podría variar significativamente frente a las crisis generadas por los problemas de recolección que existen anualmente.

Twitter

La identificación de tweets útiles, según se definen en la fórmula (4.1), plantea algunas dificultades que pueden complejizar la solución. Por mencionar algunas:

- La dificultad de identificar la ciudad sin que el texto tenga referencias explícitas y sin que el tweet esté georreferenciado (Figura 4.3)
- La dificultad de distinguir entre un tweet que trata a la limpieza como una noticia (Figura 4.4), de uno que es un reclamo (Figura 4.1).
- La presencia de la temática en las imágenes adjuntas y no en el texto (Figura 4.5).

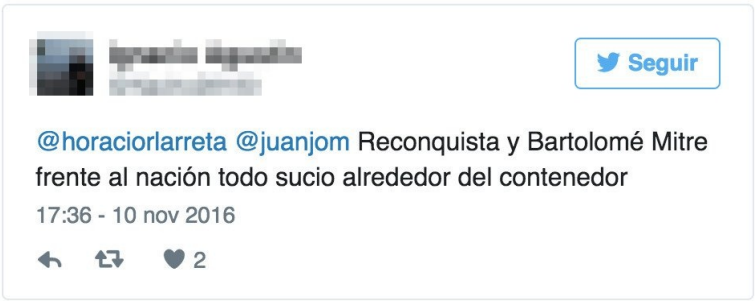


Figura 4.3: Tweets sobre limpieza en Buenos Aires

Figura 4.4: Tweets sobre limpieza como noticia

Figura 4.5: Tweet sobre limpieza expresada en las imágenes adjuntas

Capítulo 5

Solución propuesta

En este capítulo se describen las principales tecnologías utilizadas, se hace una descripción general de la arquitectura de la solución y finalmente se especifica en detalle la construcción de cada uno de los módulos que componen la plataforma.

5.1. Herramientas utilizadas

En este trabajo se utilizaron diferentes tecnologías buscando favorecer herramientas *open source* y aquellas que presentaran un desafío de aprendizaje para el equipo. En esta sección se describen las características que fundamentan su elección para este trabajo. Además se presentan los diferentes servicios de terceros utilizados en el desarrollo de la solución.

5.1.1. AngularJS

AngularJS[1] es un *framework* de JavaScript de código abierto, mantenido por Google, para aplicaciones web dinámicas. Se ejecuta directamente en el navegador por lo que es independiente de la tecnología que se decida utilizar para el servidor web.

Permite la creación de aplicaciones web de calidad de una forma rápida y sencilla.

Se utilizó como *frontend* para los módulos de anotación de *corpus* y visualización de resultados.

5.1.2. NodeJS

NodeJS[2] es un entorno en tiempo de ejecución de Javascript, de código abierto y mulitplataforma. Es utilizado para la creación de aplicaciones de red de forma rápida y escalable.

Las aplicaciones en NodeJS se escriben en Javascript y se pueden ejecutar en cualquier sistema operativo. Proporciona además una gran cantidad de librerías Javascript que permiten agilizar la construcción de aplicaciones web.

Se utiliza como servidor para los módulos web de este trabajo.

[1] https://angularjs.org/
[2] https://nodejs.org/

5.1.3. Scikit-Learn

Scikit-Learn[3] es una librería de código abierto para el Aprendizaje Automático en el lenguaje Python.

Scikit-Learn cuenta con algoritmos llamados "transformadores"que convierten los documentos en representaciones como *bag of words*, *term vectors*.

Además cuenta con diferentes "estimadores"que implementan gran variedad de algoritmos de clasificación, clusterización y regresión.

La facilidad de uso y la diversidad de estimadores implementados por esta librería fueron determinantes para su elección a la hora de implementar tareas de aprendizaje automático.

5.1.4. Herramientas del Grupo de PLN de Stanford

Las herramientas de Etiquetado gramatical de palabras (POS) y Reconocimiento de Entidades con Nombre (NER) del grupo de Procesamiento de Lenguaje Natural de la Universidad de Stanford, son analizadores de texto que mediante modelos pre-entrenados [61] [22] pueden etiquetar el texto de entrada con información POS y NER. Ambas están disponibles en su sitio web[4] [58].

Es posible re entrenar los modelos con diferentes *corpus* de datos según las necesidades del problema.

5.1.5. Detección de lenguajes

Langid[5] y Langdetect[6] son dos librerías de código abierto para Python que dada una entrada de texto devuelven la probabilidad de pertenencia a cierto lenguaje. Implementan métodos de clasificación pre entrenados cuya predicción es veloz. Combinadas, ambas librerías pueden usarse para determinar si un texto pertenece a un determinado idioma. Se utiliza la detección de lenguajes para construir una *feature* binaria utilizada en uno de los clasificadores.

5.1.6. MongoDB

MongoDB[7] es un motor de bases de datos no relacional open source, orientado a documentos. Los documentos, compuestos por pares clave-valor, son almacenados como BSON, un formato binario similar a JSON [43].

5.1.6.1. MongoDB y su flexibilidad

MongoDB permite gran flexibilidad al carecer de restricciones de estructura. En una misma colección pueden coexistir documentos con estructuras diferentes permitiendo la actualización y el agregado de datos a los documentos sin restricciones.

Según la arquitectura definida, a medida que una Pieza de Información cuyo origen es un tweet atraviesa las distintas fases, es enriquecida con diferentes tipos de datos en

[3]http://scikit-learn.org/stable/
[4]http://nlp.stanford.edu/software/
[5]https://github.com/saffsd/langid.py
[6]https://github.com/Mimino666/langdetect
[7]https://www.mongodb.com/es

diferentes momentos por cada uno de los módulos involucrados. Gracias a la elección de MongoDB estos cambios en las Piezas de Información se hacen en la misma colección y de forma concurrente sin dificultades.

5.1.6.2. MongoDB y datos geográficos

MongoDB introduce el formato GeoJSON, utilizado para almacenar estructuras geoespaciales en los documentos. Cada elemento GeoJSON está compuesto por:

- Coordinates: una lista de coordenadas que representa los vértices de la geometría.
- Type: campo que indica el tipo de geometría y sirve para interpretar el campo coordinates.

MongoDB soporta índices geoespaciales sobre campos de tipo GeoJSON. Estos índices, cuando existen, permiten la realización de consultas geográficas a través de sus operadores:

- *Near*: dado un objeto de tipo GeoJSON y especificando distancias mínimas y máximas, permite obtener los documentos cercanos al objeto ordenados de más cercanos a más lejanos.
- *Intersects*: dado un objeto de tipo GeoJSON permite obtener los objetos que se intersectan geográficamente con este objeto.

5.1.6.3. MongoDB y búsquedas textuales

MongoDB permite la creación de índices sobre los campos de texto. Estos índices tienen características muy útiles en búsquedas en campos de texto con lenguaje natural:

- *Stopwords*: la búsqueda permite ignorar las *stopwords* del lenguaje especificado.
- *Stemmed Words*: la comparación de términos para la búsqueda se puede realizar sobre los lemas de las palabras.
- *Case Insensitivity*: es posible hacer la comparación de términos ignorando mayúsculas.
- *Diacritic Insensitivity*: es posible hacer la comparación ignorando acentos.

5.1.7. Docker

Docker[8] es una implementación de contenedores Linux (LXC). Permite el despliegue de aplicaciones en entornos virtualizados junto a sus librerías y dependencias con mínimo overhead en los llamados contendores.

Docker facilita la replicación de ambientes en entornos de desarrollo y producción así como el despliegue de aplicaciones de forma rápida y escalable.

Este trabajo integra varias tecnologías en diferentes módulos que interactúan entre sí, compartiendo datos y servicios. La aislación que permite Docker de cada uno de los módulos y la facilidad para ejecutar la plataforma completa justifican la elección de esta herramienta.

Como ventaja adicional permite la rápida reproducción de resultados obtenidos por terceros que sería de otro modo extremadamente engorrosa [6].

[8] https://www.docker.com/

5.1.8. Twitter Streaming API

API provista por Twitter[9] que permite, dado un criterio de búsqueda, obtener tweets en directo que cumplan con este citerio. Esta API permite evitar el *overhead* asociado a hacer *polling* en busca de nuevos tweets.

5.1.9. Pentaho Data Integration

Pentaho Data Integration[10] es una herramienta open-source utilizada para la construcción de procesos de extracción, transformación y carga de datos (ETL).

Fue utilizada durante el preprocesamiento de uno de los *corpus* utilizados.

5.1.10. Python Pandas

Python Pandas[11] es una librería open-source que provee herramientas de fácil uso y alta performance para el análisis de datos estructurados.

Al igual que Pentaho es utilizada durante el preprocesamiento.

5.1.11. Corpus General Tass 2015

Tass (Taller de Análisis de Sentimientos)[12] son talleres experimentales alrededor de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN)[13]. Anualmente se proponen en estos talleres tareas para la evaluación del estado del arte en Procesamiento del Lenguaje Natural (PLN). Mediante un pedido vía mail especificando motivos académicos, es posible solicitar la descarga del *corpus* de estos talleres.

El *corpus* cuenta con aproximadamente 60 mil tweets en español escritos por diferentes personalidades influyentes de diversos ámbitos y países. La temática de los tweets abarca política, fútbol, literatura [26].

5.1.12. Flask

Flask[14] es un *framework* de Python para la implementación de servicios rest de forma sencilla. Es el *framework* utilizado para exponer las API rest utilizadas como *backend* de las aplicaciones web.

5.1.13. Leaflet

Leaflet[15] es una librería Javascript open-source que permite obtener mapas interactivos de forma sencilla.

Es la librería elegida para visualizar los Eventos georreferenciados sobre el mapa de la ciudad.

[9] https://dev.twitter.com/streaming/overview
[10] http://community.pentaho.com/projects/data-integration/
[11] http://pandas.pydata.org/
[12] http://www.sepln.org/workshops/tass/2015/tass2015.php#corpus
[13] http://www.sepln.org
[14] http://flask.pocoo.org/
[15] http://leafletjs.com

5.1.14. Open Street Map

Open Street Map (OSM)[16] es un proyecto colaborativo para la creación de mapas libres y editables. Además es el proveedor del que se alimenta Leaflet.

5.1.15. Google Cloud Vision API

Google Cloud Vision API[17] es un servicio provisto por Google que permite el procesamiento de imágenes, encapsulando poderosos modelos de Inteligencia Artificial (IA) en una API rest.

5.1.16. IBM Visual Recognition

IBM Visual Recognition[18] es un servicio provisto por IBM que permite el procesamiento de imágenes.

5.1.17. Python Keras

Python Keras[19] es una librería de Python creada para la implementación rápida de modelos de IA.

Junto a Google Cloud Vision API e IBM Visual Recognition se utiliza en el módulo de reconocimiento de imágenes.

5.2. Arquitectura de la solución

En esta sección se presenta la arquitectura de una plataforma que permita la recuperación de tweets generados por ciudadanos de Montevideo referentes a la limpieza de la ciudad y su posterior visualización para su mejor gestión.

5.2.1. Descripción de la arquitectura

Según se vio en la Figura 4.2 lo que se busca determinar para cada nuevo tweet analizado por la plataforma es su pertenencia al conjunto de los "Tweets útiles" definido por la fórmula (4.1).

Los "Tweets útiles", según las elecciones de dominio y ciudad para este trabajo, corresponden a tweets que son reclamos en español, que tratan sobre limpieza y son de Montevideo.

Desde que un tweet es generado por un ciudadano hasta convertirse en un Evento para ser analizado por los usuarios del sistema se recorren cuatro etapas:

- **Etapa 1:** Recolección de la información.
- **Etapa 2:** Enriquecimiento de la información.
- **Etapa 3:** Clasificación de la información y generación de Eventos.

[16] www.openstreetmap.org
[17] https://cloud.google.com/vision
[18] https://www.ibm.com/watson/developercloud/visual-recognition.html
[19] https://keras.io/

- **Etapa 4:** Visualización de los Eventos.

A continuación se explica brevemente en qué consiste cada una de las etapas que componen el sistema como una forma de introducción a la solución planteada.

Etapa 1: Recolección de información

En primer lugar la plataforma debe recuperar y almacenar todos aquellos tweets candidatos a ser "Tweets Útiles" para la ciudad y domino elegidos.

Para la recolección de los tweets se utilza la Streaming API de Twitter. El criterio de búsqueda utilizado fue creado a partir de la combinación de dos listas de palabras:

- Palabras claves de Montevideo (Montevideo, im, imm, @monteideoim, etc.).
- Palabras claves de limpieza (contenedor, recolector, papelera, basura, etc.).

Cada uno de los tweets recolectados se almacenan en una base de datos MongoDB. Al almacenar la información obtenida por la API se descartan algunos campos del *JSON* obtenido. Estos campos no agregan ninguna información relevante para este trabajo y se decidió descartarlos como forma de minimizar el espacio de almacenamiento de la información recolectada.

Los tweets almacenados constituyen una Pieza de Información que será enriquecida en la Etapa 2.

A los efectos de este trabajo una Pieza de Información es un documento que incluye algunos de los datos de la fuente de información original y algunos campos de control que son utilizados por la Etapa 2 en el flujo definido por la plataforma.

Etapa 2: Enriquecimiento de la información

En esta etapa actúan varios módulos que componen el sistema cuya tarea es enriquecer la información obtenida en la Etapa 1. Cada módulo responde a una estrategia para determinar si un tweet pertenece a conjuntos de tweets que interesan al problema (tweets sobre limpieza, tweets de Montevideo, tweets con reclamos en español). Los módulos aportan nuevas características que enriquecen la información original y ayudan a determinar en la Etapa 3 si una Pieza de Información corresponde a un Evento o no.

Los módulos que participan de la Etapa 2 son los siguientes:

- El **Módulo de información gramatical** enriquece el texto de las Piezas de Información con datos de roles gramaticales y entidades con nombre.
- El **Módulo de georreferenciación** busca procesar el texto de cada una de las Piezas de Información en busca de una ubicación específica dentro de Montevideo, o descartar aquellos que sean originados en ubicaciones distintas.
- El **Módulo de procesamiento de imágenes** intenta decidir si las imágenes adjuntas a cada Pieza de Información poseen elementos asociados a basura.
- El **Módulo clasificador de reclamos** busca determinar si una Pieza de Información corresponde a un reclamo en español, basándose en su texto.
- El **Módulo clasificador de limpieza** asigna una probabilidad a partir del texto, para determinar si se hace referencia al dominio del problema: la limpieza.

En la Sección 5.3 se explica más en detalle cada uno de los módulos que componen esta etapa, y su funcionamiento. En el Capítulo 6 se muestra el desempeño de cada uno individualmente.

Etapa 3: Clasificación de la información y generación de eventos

En la Etapa 3, la plataforma debe decidir si una Pieza de Información corresponde a un Evento o no. Cada uno de los elementos incorporados en la Etapa 2 serán insumos para esta decisión.

Esta etapa consta únicamente del **Modulo de detección de Eventos** cuya tarea es filtrar y descartar aquellas Piezas de Información que no sean Eventos y almacenar aquellas que sí lo sean en una nueva colección para su posterior visualización y gestión en la Etapa 4.

En la Sección 5.3 se presenta en más detalle la construcción y el funcionamiento del módulo y en el Capítulo 6 se muestran los resultados obtenidos.

Etapa 4: Visualización de los eventos

En la etapa final del sistema se recuperan los Eventos detectados en la Etapa 3 para su visualización por parte del usuario. Para ello se construyó una web [20] en la que se muestra el mapa de Montevideo junto con los Eventos detectados marcados según su ubicación, la que fue obtenida por el Módulo de georreferenciación de la Etapa 2, como se muestra en la Figura 5.1.

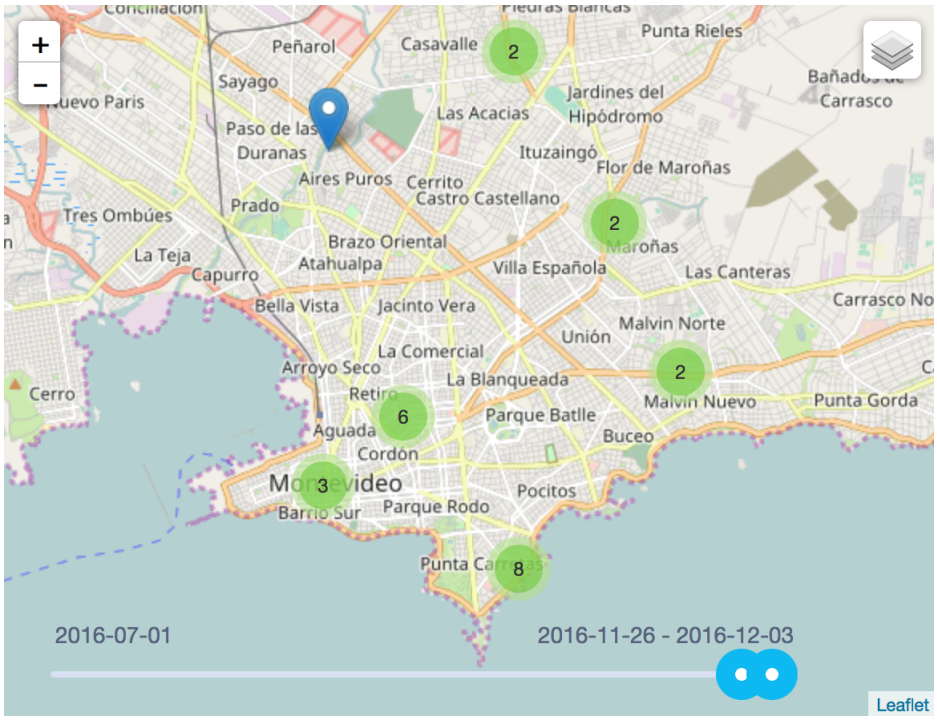


Figura 5.1: Página principal de la web donde se muestran los Eventos detectados

En la Sección 5.3 se explica más en detalle la visualización de los diferentes Eventos detectados por el sistema.

[20] https://krypton.mgcoders.uy

5.2.2. Diseño de la arquitectura

En 5.2.1 se describieron las diferentes etapas involucradas en la solución. Como se puede ver en la Figura 5.2 una Pieza de Información recorre una serie de módulos cuyo funcionamiento se detalla en la Sección 5.3.

La implementación de cada uno de los módulos se realizó en el lenguaje Python, la persistencia de datos se hizo utilizando MongoDB y las páginas web se desarrollaron en AngularJS y NodeJS.

Disminuir el tiempo entre que un tweet es realizado por un ciudadano y que es procesado o descartado como Evento por la plataforma es uno de los aspectos relevantes si se tiene en cuenta el valor que los usuarios le dan a la inmediatez en Twitter. Además, dependiendo del volumen de información que ingrese al sistema es importante poder escalar y/o distribuir los módulos para que la performance no se vea afectada.

Se diseñó una arquitectura que permite la ejecución independiente, replicada y distribuida de cada módulo. La Figura 5.3 muestra un ejemplo de ejecución donde el Módulo 1 tiene $\boldsymbol{n}$ instancias en ejecución, y donde cada uno de los módulos puede correr en servidores independientes. Este modelo permite la incorporación, eliminación y cambio de un módulo con mínimo impacto en los demás.

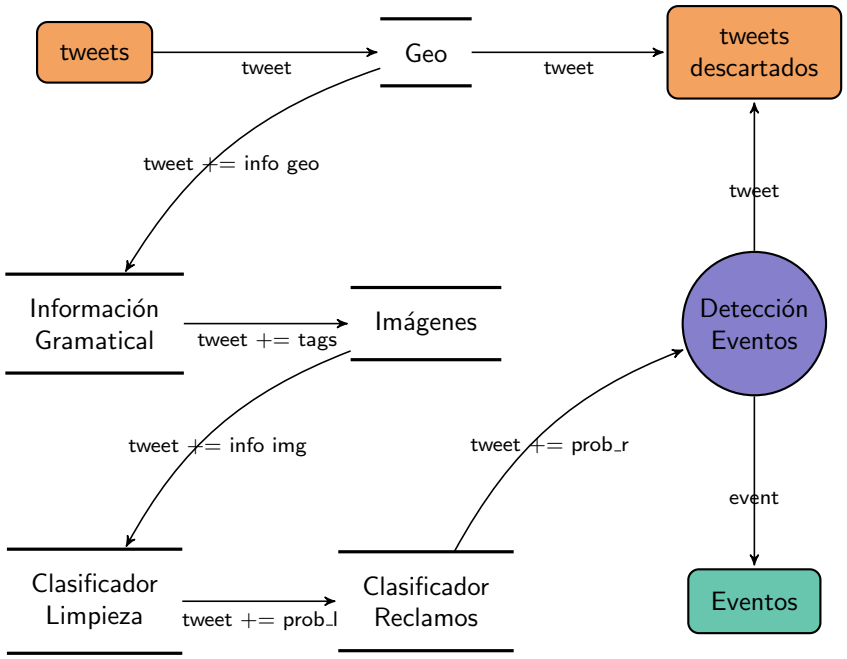


Figura 5.2: Interacción entre módulos

Los datos se almacenan como documentos en formato JSON y el pasaje por cada módulo se guarda en un documento usando campos específicos que permiten reflejar al resto del sistema, el estado de una Pieza de Información y su posición en el flujo planteado.

Como se explica en 5.1.7, Docker fue la tecnología elegida para el despliegue y distribución del software. Esta herramienta permite que cada módulo (normalmente dependiente de complejas librerías y configuraciones) se distribuya con sus dependencias y se pueda ejecutar en un espacio de procesos aislado sin comprometer recursos de memoria o CPU. Docker posibilita escalar horizontalmente permitiendo la ejecución de varias copias de

un mismo módulo y es sencillo distribuir los módulos en distintos servidores según sea necesario.

La arquitectura diseñada y el modelo de datos elegido permite que la solución tenga las características mencionadas en la Sección 4.3, esto es: una plataforma instanciable en varios dominios y ciudades, extensible mediante módulos y compatible con otros sistemas.

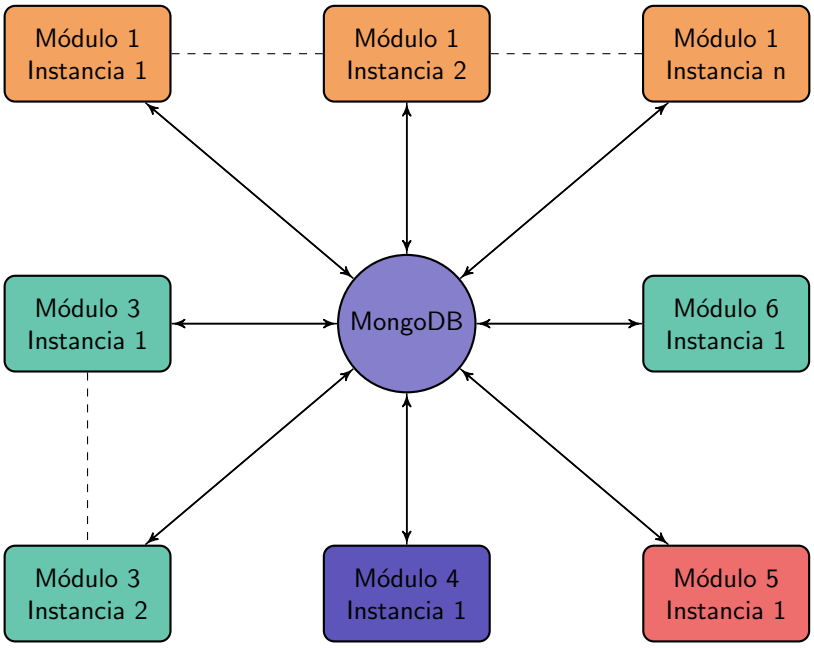


Figura 5.3: Diseño de Arquitectura

5.3. Diseño de la solución

En esta sección se detalla el diseño y funcionamiento de cada uno de los módulos involucrados en la plataforma desarrollada. En línea con lo explicado en la Sección 4.3, se ha buscado en cada módulo una implementación lo más abstracta posible de manera de que la plataforma resultante pueda ser instanciada en otro dominio o en otra ciudad de la forma más sencilla. En la parte correspondiente a la descripción de cada módulo se detalla de qué manera la implementación es generalizable para el dominio y ciudad elegidos.

5.3.1. Módulo de extracción de tweets

El objetivo del Módulo de extracción de tweets es recuperar y almacenar durante la Etapa 1 todos los tweets que puedan ser de interés para el problema con la menor demora posible. Los resultados de este módulo se pueden ver en la Sección 6.1.

5.3.1.1. Descripción

Como se explicó en la Sección 4.1, según el alcance definido para este trabajo se utiliza como fuente de datos la red social Twitter. La extracción de tweets se realiza mediante

la Streaming API de Twitter que permite obtener tweets casi en tiempo real filtrados por frases y palabras clave o por origen geográfico.

El filtrado por origen geográfico utiliza dos pares de coordenadas que delimitan el rectángulo que rodea al departamento de Montevideo. Al activar esta opción, la API retorna cualquier tweet originado dentro de esa área sin tomar en cuenta las palabras clave. Es importante destacar que según mediciones obtenidas de [39], en un día promedio solo el 2.02 % de los tweets incluyen información geográfica, 1.8 % tiene referencia de alguna ciudad, 1.6 % tiene ubicación exacta y el 1.4 % tiene ambas, por lo que se optó por no activar esta opción en primera instancia.

Para el filtrado por palabras clave se utilizan dos listas elaboradas experimentalmente a partir de diferentes pruebas. Una de las listas contiene palabras clave referentes a la ciudad de Montevideo, como pueden ser las denominaciones habituales de la ciudad o cuentas de organismos municipales y autoridades (Tabla 5.1). La otra lista contiene palabras clave referentes al dominio elegido: contenedor, limpieza, etc., (Tabla 5.2). Ambas listas son combinadas de manera de obtener tweets que tengan al menos una palabra de cada lista.

Ciudad
im
imm
@montevideoim
intendencia montevideo
montevideo
@quejasya
@quejatedetodo
@MunicipioE
@MunicipioGMvd
@municipioa
@Municipiob

Tabla 5.1: Algunas palabras clave de ciudad

Dominio
contenedor
contenedores
recolector
recolectores
escombros
desbordado
papelera
tacho
basura
residuo
bolsas

Tabla 5.2: Algunas palabras clave de dominio

Las API públicas y gratuitas de Twitter tienen límites establecidos para la cantidad de palabras clave que se pueden utilizar. Existen otros servicios de pago que permiten la

descarga ilimitada de tweets a los suscriptores. En un escenario no académico la extracción de tweets podría realizarse usando combinaciones de todas las palabras más frecuentes del español asegurando que la plataforma no pierda ningún dato que puede terminar siendo relevante para el dominio y la ciudad.

La extracción de tweets para este trabajo comenzó en las etapas iniciales del proyecto con diferentes pruebas y mecanismos. Una vez definido el mecanismo definitivo, es decir la extracción mediante la combinación de listas de palabras clave, se procedió a activar el módulo de forma permanente. El tweet más antiguo almacenado es del 27-09-2016 y el más nuevo del 10-01-2017. Todos los resultados y métricas de este módulo para ese período se pueden ver en la Sección 6.1.

En un intento por contar con un *corpus* más extenso de tweets que cumplieran con los criterios de los descargados mediante este módulo, se hicieron averiguaciones con servicios que, a diferencia de las API de Twitter, permiten descargar tweets de períodos anteriores al actual. El elevado costo de estos servicios hizo que se descartara su utilización.

Cada tweet al ser recuperado, es almacenado como una Pieza de Información manteniendo una estructura común con otras fuentes de datos que puedan existir. Las Piezas de Información mantienen el texto original, la ubicación, si existe, y una serie de metadatos para ser utilizados posteriormente por la plataforma.

Los tweets pueden ser retweets (un mecanismo para compartir tweets ajenos), pueden ser respuestas a otros tweets, o pueden tener otro tweet adjunto. En estos casos se recuperan también los tweets relacionados como se muestra en la Figura 5.4

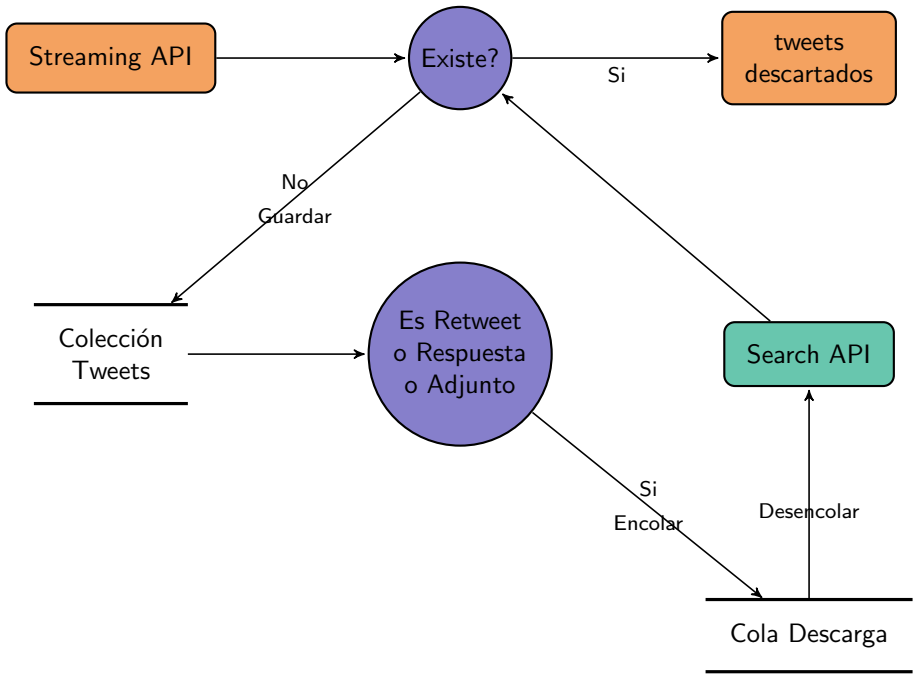


Figura 5.4: Mecanismos de descarga de Tweets

5.3.1.2. Generalización

La implementación de este módulo es completamente independiente del dominio y ciudad elegidos. El módulo se basa en listas de palabras almacenadas en archivos de texto

que pueden ser modificados en cualquier momento.

5.3.2. Módulo de anotación del corpus

El objetivo de este módulo auxiliar es la anotación manual de los tweets recolectados por el Módulo de extracción de tweets (5.3.1), para su utilización durante los procesos de obtención de resultados y para el entrenamiento del Módulo de detección de Eventos (5.3.8).

5.3.2.1. Descripción

Para la anotación del *corpus* de tweets se tomaron en cuenta cuatro etiquetas, las que dependen tanto del texto del tweet como de las imágenes adjuntas. Las etiquetas utilizadas fueron:

- **MUY_UTIL**: Se refiere a la limpieza en Montevideo y contiene una ubicación específica. En la Figura 5.5a se puede observar un ejemplo de tweet que se anotó con la etiqueta MUY_UTIL.
- **UTIL**: Se refiere a la limpieza en Montevideo y contiene alguna referencia de ubicación. Un ejemplo de tweet que se anotó con la etiqueta UTIL se puede ver en la Figura 5.5b.
- **POCO_UTIL**: Se refiere a la limpieza en Montevideo, sin referencia de ubicación. En la Figura 5.5c se puede ver un ejemplo de tweet con esta etiqueta.
- **NADA_UTIL**: No se refiere a la limpieza en Montevideo. Un ejemplo de tweet perteneciente a esta etiqueta puede verse en la Figura 5.5d

En la Sección 6.1 se especifican las cantidades obtenidas para cada etiqueta.

(a) Ejemplo de tweet MUY_UTIL

(b) Ejemplo de tweet UTIL

(c) Ejemplo de tweet POCO_UTIL

(d) Ejemplo de tweet NADA_UTIL

Figura 5.5: Ejemplos de tweets según etiqueta

Se escogió utilizar una escala para la anotación ya que permite más flexibilidad a la hora de elegir los criterios para definir si un tweet es relevante o no para el dominio del proyecto.

Para la realización de este módulo se desarrolló una web que se encuentra alojada en `http://krypton.mgcoders.uy/classify/`. Para la implementación de este módulo se utilizó como ya se mencionó, AngularJS y NodeJS.

Como se mencionó en 5.3.1 las Piezas de Información se almacenan en una base de datos MongoDB por lo que se decidió implementar un servidor web, utilizando Python, para que realice la comunicación entre la web y la base de datos MongoDB. Este servidor web publica una API rest que es consumida por la web desarrollada. Para la implementación de la API se utilizó la librería Flask.

El sitio web se compone de dos páginas: página de inicio de sesión y página principal.

En la página de inicio de sesión, además de ingresar las credenciales para ingresar a la plataforma, se puede crear un nuevo usuario. Para la creación de usuarios es necesario la introducción de un correo electrónico, una contraseña y una palabra clave que es parametrizable. Los usuarios, junto con sus credenciales cifradas, son almacenados en una base de datos MongoDB y la comunicación entre la web y la base se realiza a través de la API rest desarrollada por el equipo.

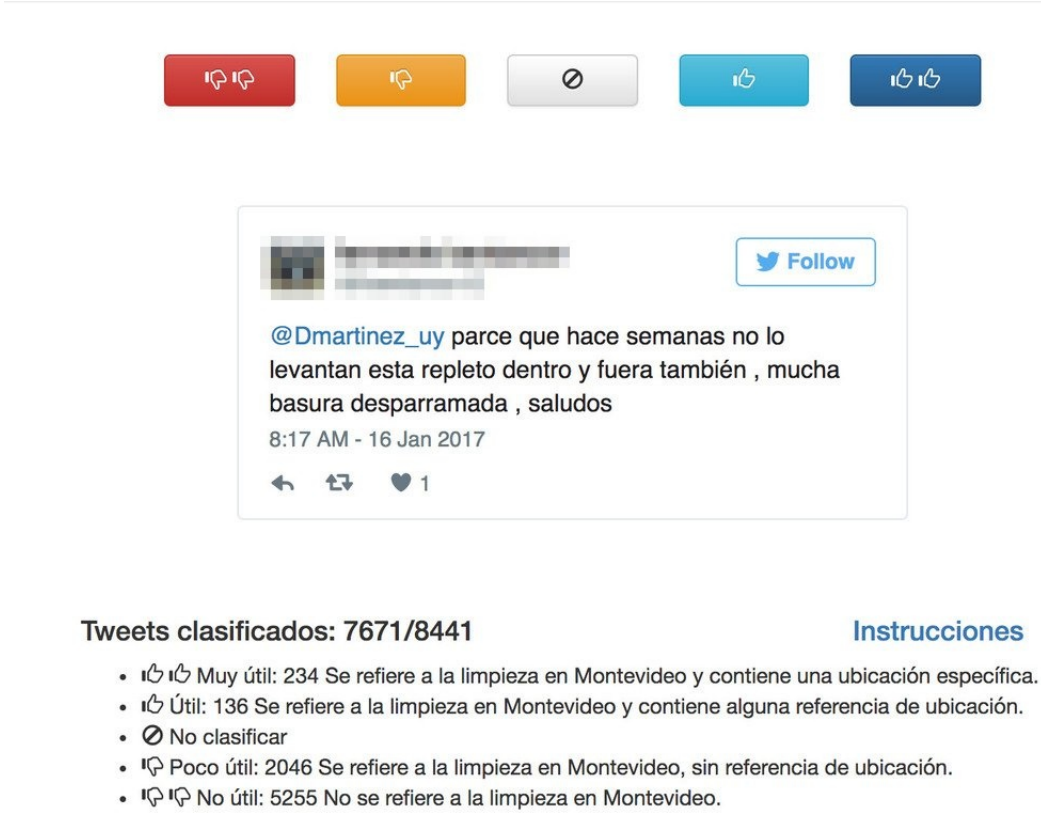


Figura 5.6: Web desarrollada para la anotación de los tweets recuperados

La página principal es la utilizada para la anotación de los tweets, se puede observar en la Figura 5.6. En ella se pueden apreciar cuatro partes:

- Botones: los botones son utilizados para anotar el tweet con la etiqueta o para saltear (botón blanco) un tweet sin anotarlo.
- Tweet: se muestra, en la parte central de la página, el tweet que se va a anotar. Para ello se utiliza una librería Javascript que permite visualizar el tweet de la misma forma en la que se ve en Twitter.

- Estadísticas: se muestra una estadística para poder, de una forma sencilla, saber cuántos tweets fueron recolectados y cuántos de ellos ya se encuentran anotados.
- Instrucciones: indica a qué corresponde cada una de las posibles etiquetas para anotar un tweet. Junto a cada una de las etiquetas se muestra la cantidad de tweets que fueron anotados con esa etiqueta.

Una vez clasificado (o salteado) un tweet, a través de la API y de forma aleatoria se obtiene uno nuevo para ser anotado. En caso de clasificarse, la información es enviada por medio de la API desarrollada, guardándose la etiqueta elegida. Como respuesta a la clasificación el servidor web envía un mensaje indicando éxito o error según corresponda.

Todas las comunicaciones que se realizan son de forma segura utilizando el protocolo HTTPS.

5.3.2.2. Generalización

Este módulo auxiliar se puede utilizar para anotar un *corpus* de tweets con las etiquetas definidas en cualquier dominio o ciudad, sin necesidad de modificaciones.

5.3.3. Módulo de información gramatical

El objetivo de este módulo es agregar a cada Pieza de Información datos sobre el rol gramatical de las palabras en el texto. En particular se busca un etiquetado gramatical de cada una de las palabras (POS) y un reconocimiento de las entidades con nombre que puedan estar presentes (NER).

5.3.3.1. Descripción

Se decidió utilizar las soluciones del Grupo de Procesamiento de Lenguaje Natural de la Universidad de Stanford para el etiquetado POS y NER presentadas en el Apéndice A.

El modelo para español de los analizadores de la Universidad de Stanford para el etiquetado NER, está entrenado sobre el *corpus* Ancora [60] y según observaciones sobre tweets descargados para este trabajo, el etiquetado de entidades con nombre solo tiene un buen desempeño en texto correctamente escrito y capitalizado.

Por lo anterior, se decidió recolectar y anotar un *corpus* de tweets a los efectos de re entrenar el modelo de Stanford, de manera de obtener un etiquetado adecuado a la estructura y el estilo coloquial de los tweets. Se desarrolló una herramienta web para anotar el *corpus* y una API para re entrenar el modelo de Stanford de forma dinámica a partir de archivos de texto tabulados.

Lamentablemente debido a la escasez de datos y la dificultad para la anotación de cada una de las entidades con nombre, el nuevo modelo dio peores resultados que el original por lo que se descartó su uso.

La motivación inicial del módulo consistía en identificar las entidades correspondientes a ubicaciones geográficas dentro del texto. Solo estas entidades serían la entrada para el Módulo de georreferenciación, de manera de acotar la búsqueda en las colecciones de datos georreferenciados. El modelo original de Stanford para NER solo es capaz de reconocer ciudades y países, pero no calles o ubicaciones más especificas.

En 5.3.4.3 se estudia la complejidad del algoritmo principal del Módulo de georreferenciación resultando estar directamente relacionada con la cantidad de palabras o *tokens* que el módulo recibe como entrada.

Tratándose de tweets, que como máximo tienen una longitud de 140 carácteres, los tiempos de ejecución del módulo de georreferenciación son aceptables aún si todas las palabras están involucradas en la búsqueda. Sin embargo resulta claro que si hubiera otras fuentes de información involucradas la performance se vería seriamente afectada. Es en ese contexto que poder contar con el correcto etiquetado POS y NER sería una gran ventaja.

La imposibilidad de anotar un *corpus* suficientemente grande como para que los modelos de Stanford fueran sensibles al estilo de los tweets llevaron a la decisión de no utilizar los resultados de este módulo en ninguna de las etapas de la identificación de Eventos.

En el Apéndice A se describen las herramientas desarrolladas para la anotación de un nuevo *corpus* y el posterior entrenamiento del etiquetador de Stanford con este *corpus*. El código se encuentra disponible en GitHub[21] como aporte a otros investigadores a quienes pueda ser útil.

5.3.3.2. Generalización

Este modulo, de utilizarse, es completamente independiente de la ciudad o dominio elegidos.

5.3.4. Módulo de georreferenciación

El objetivo del Módulo de georreferenciación es enriquecer duarante la Etapa 2 una Pieza de Información con datos geográficos precisos de acuerdo al dominio y a la ciudad elegidos.

Los datos de entrada del módulo son el texto de la pieza, y opcionalmente datos geográficos, en el caso de Twitter la ubicación de origen del tweet. La salida es la Pieza de Información enriquecida con una lista de posibles soluciones ordenadas según un puntaje asignado. Cada posible solución es un dato geográfico encontrado como ubicación a partir de los datos de entrada.

En la Sección 6.2 se discuten los resultados del módulo para ciertos ejemplos y ciertas métricas de desempeño.

5.3.4.1. Fuentes de Información

Este módulo requiere de dos tipos de fuentes de información georreferenciada. Por un lado la información de interés sobre la ciudad elegida y por otro la información referente al dominio sobre el que se trabaja. Se utilizaron varias colecciones de datos abiertos sobre la ciudad de Montevideo según se detalla en la Tabla 5.3. Los datos fueron descargados y están disponibles en el portal de datos abiertos del Estado[22].

5.3.4.2. Algoritmo

En primer lugar el algoritmo descarta aquellos datos de entrada que de acuerdo a los datos geográficos fueron originados en una ciudad diferente a la elegida. Sería descartado en este momento un tweet con "Paysandú" o "Buenos Aires" en el campo ubicación.

Si el dato no puede ser descartado se realiza una separación en *tokens* y preprocesamiento del texto de entrada. El preprocesamiento consiste en eliminanr *stopwords*,

[21] https://github.com/raulsperoni/stanford-taggers-tools
[22] http://datos.gub.uy/

Información	Tipo
Vías de tránsito	Ciudad
Cruces de calle	Ciudad
Espacios libres	Ciudad
Puntos de interés	Ciudad
Límites de barrios	Ciudad
Contenedores de residuos reciclables	Dominio
Contenedores de residuos secos	Dominio
Equipamiento urbano, papeleras	Dominio

Tabla 5.3: Colecciones de datos abiertos utilizadas

Ubicación encontrada	Puntaje
Calle con calle	80
Calle con lugar	40
Calle con espacio	40
Lugar con calle	50
Lugar con espacio	40
Barrio	10

Tabla 5.4: Puntajes asignados a cada uno de los elementos encontrados

símbolos no alfanuméricos y palabras muy frecuentes en nombres de calles y espacios públicos. Se procesan además los *hashtags* separando palabras.

Para cada *token* se obtienen listas de calles, espacios públicos, puntos de interés y barrios que estén relacionados. Esta búsqueda utiliza índices de texto con *stemming* de la base de datos. Luego se ejecutan estrategias de búsqueda que combinan las diferentes listas mediante consultas geográficas para encontrar intersecciones de calles, intersecciones de lugares con barrios, puntos de cercanía entre lugares y calles, etc. El funcionamiento de los índices textuales y geográficos de MongoDB se detalla en 5.1.6.

Cada intersección encontrada constituye una posible solución a la que corresponde un puntaje según cómo haya sido encontrada.

A modo de ejemplo, se considera más relevante una intersección entre calles que un punto de cercanía entre una calle y un lugar.

Aumenta la relevancia de una solución si otros elementos del texto confirman su validez, como puede ser la presencia del nombre del barrio correspondiente. Esta relevancia se expresa en el puntaje asignado.

Para cada una de las soluciones se obtiene un *geohash* que permite descartar soluciones que son iguales tomando como criterio cierta cercanía espacial. Un *geohash* es una codificación de coordenadas geográficas cuya representación es una palabra y básicamente representa un rectángulo en la superficie de la tierra. El tamaño del rectángulo, y por lo tanto la precisión de la ubicación viene dado por la longitud de la palabra [23]. Se utilizó una longitud de ocho caracteres lo que permite agrupar soluciones que están aproximadamente a 20 metros entre sí.

Finalmente se buscan elementos del dominio (contenedores, papeleras) cercanos geográficamente a cada solución. Los elementos encontrados y la distancia también aportan al puntaje de las soluciones que, ordenadas, son la salida del módulo.

En la Tabla 5.4 se muestran los puntajes asignados para cada uno de los elementos

```
1: function procesamientoGeo(text, coordinates)
2:     soluciones ← ∅
3:     if ubicacionFueraDeCiudad(coordinates) then
4:         return soluciones
5:     end if
6:     tokens ← tokenizar(text)
7:     tokens ← eliminarStopwors(tokens)
8:     tokens ← eliminarTerminosComunes(tokens)
9:     barrios ← ∅
10:    lugares ← ∅
11:    espacios ← ∅
12:    calles ← ∅
13:    for token in tokens do
14:        barrios ← barrios + obtenerBarrios(token)
15:        lugares ← lugares + obtenerLugares(token)
16:        espacios ← espacios + obtenerEspacios(token)
17:        calles ← calles + obtenerCalles(token)
18:    end for
19:    geoHash ← ∅
20:    soluciones ← soluciones + porCoordenadas(coordinates, geoHash)
21:    soluciones ← soluciones+porLugar(calles, lugares, espacios, barrios, geoHash)
22:    soluciones ← soluciones+porCalle(calles, lugares, espacios, barrios, geoHash)
23:    for solucion in soluciones do
24:        solucion ← buscarContenedor(solucion)
25:        solucion ← buscarContenedorResiduos(solucion)
26:        solucion ← buscarPapeleras(solucion)
27:    end for
28:    soluciones ← ordenarPorPuntaje(soluciones)
29:    return soluciones
30: end function
```

Figura 5.7: Algoritmo para extraer información geográfica

encontrados. Adicionalmente se suma al puntaje final una cantidad x que se define como en la fórmula 5.1. Cada elemento corresponde a un tipo de mobiliario urbano y la constante C asociada permite asignar la importancia de cada tipo. $C_{contenedor} = 1000$, $C_{residuo} = 800$ y $C_{papelera} = 600$.

$$x = \frac{C_{elemento}}{dist(solucion, elemento)} \tag{5.1}$$

```
1: function BUSCARSOLUCIONESCONCALLE(calles, lugares, espacios, barrios)
2:     soluciones ← ∅
3:     for calle in calles do
4:         solucion ← ∅
5:         solucion ← BUSCARINTERSECCIONCONCALLES(calle, solucion)
6:         solucion ← BUSCARINTERSECCIONCONBARRIOS(calle, solucion)
7:         solucion ← BUSCARINTERSECCIONCONLUGARES(calle, solucion)
8:         solucion ← BUSCARINTERSECCIONCONESPACIOS(calle, solucion)
9:     end for
10:    solucion ← CALCULARPUNTAJE(solucion)
11:    solucion ← CALCULARGEOHASH(solucion)
12:    noExisteSolucion ← NOEXISTESOLUCIÓNSEGÚNGEOHASH(solucion)
13:    if noexisteSolucion then
14:        soluciones ← soluciones + solucion
15:    end if
16:    return soluciones
17: end function
```

Figura 5.8: Algoritmo para obtener soluciones a partir de una calle

En las Figuras 5.7 y 5.8 se puede ver un pseudocódigo del algoritmo utilizado.

5.3.4.3. Complejidad

La complejidad del algoritmo presentado se puede calcular de la siguiente manera: Sea N la cantidad de *tokens* , sean L, C, E, B la cantidad de Lugares, Calles, Espacios y Barrios obtenidos. Expresando L, C, E, B como $X_L N, X_C N, X_E N, X_B N$ respectivamente, se puede escribir el término de mayor complejidad del algoritmo como:

$$X_L N(X_C N + X_E N + X_B N) + X_C N(X_E N + X_L N + X_C N + X_B N) \simeq N^2(...) \simeq O(N^2)$$

Dado el alcance definido y la limitación de las fuentes de datos a Twitter, en este trabajo se puede esperar que N sea razonablemente bajo y por lo tanto que la demora del algoritmo con complejidad $O(N^2)$ en los peores casos sea aceptable. En 7.2.1 se discuten alternativas para mejorar la complejidad del algoritmo en escenarios donde N sea mayor.

5.3.4.4. Generalización

La implementación de este módulo quizá sea la menos generalizable a otros dominios o ciudades. Gran parte de la implementación está atada a la cantidad y diversidad de datos geográficos disponibles. Para cada una de las colecciones de datos de la Tabla 5.3 hubo que hacer un trabajo de preprocesamiento, almacenaje en bases de datos y posterior

creación de índices espaciales. A nivel de implementación cada colección interactúa de forma particular con el algoritmo dado que difieren los tipos de información geográfica (puntos, líneas o polígonos) y metadatos.

5.3.5. Módulo procesamiento de imágenes

El objetivo de este módulo es determinar si las imágenes adjuntas a un tweet pertenecen al dominio. En esta sección se describen los pasos seguidos para la implementación del módulo. Los resultados se explican en detalle en la Sección 6.3.

5.3.5.1. Descripción

El aumento en la utilización de los dispositivos móviles, se ve reflejado en el aumento de contenido multimedia en las RRSS, particularmente en Twitter.

Luego de las primeras pruebas realizadas para la búsqueda de tweets referidos a la limpieza en Montevideo se constató que muchos de ellos contienen imágenes adjuntas. Estas imágenes, como por ejemplo las de la Figura 5.9, pueden complementar lo expresado en el texto del tweet, pero también pueden ser el único método para detectar si una Pieza de Información pertenece al domino, como la de la Figura 5.10.

Por esto, se decidió desarrollar un módulo de procesamiento de imágenes que agregue información sobre el contenido multimedia adjunto en los tweets. Para ello se analizaron dos opciones:

- Utilizar servicios provistos por Google e IBM en los que se devuelven etiquetas[23] encontradas en las imágenes, como se muestra en la imagen Figura 5.11.

- Entrenar un modelo que pudiera detectar contenedores de Montevideo sin importar la cantidad de basura a su alrededor.

Como lo que se buscaba era encontrar la mayor cantidad de tweets referidos a la limpieza en Montevideo, lo primero que se intentó fue entrenar un modelo propio. Para ello fue necesario construir un *corpus* de imágenes, tal cual se explica en 5.3.5.2.

Luego de diez horas de entrenamiento sobre un *corpus* de 4000 imágenes, se obtuvo un modelo cuya precisión era similar a "lanzar una moneda", por lo que se descartó su utilización y se concluyó que los malos resultados obtenidos, entre otros factores, se debían al tamaño y la calidad del *corpus* construido.

Finalmente, dada la dificultad para construir un *corpus* de imágenes lo suficientemente grande y la falta de tiempo y recursos para realizarlo, se decidió por la opción de utilizar los servicios provistos por Google e IBM.

Si bien estos servicios no ofrecen la posibilidad de detectar contenedores de Montevideo sin basura, como muestra la Figura 5.12, sí devuelven etiquetas que pueden servir a la hora de definir si una imagen se refiere a la limpieza en general.

Evaluando ambos servicios para diferentes imágenes del dominio en las que se notaba claramente la presencia de basura los resultados fueron bien diferentes. Un ejemplo puede verse en la Figura 5.13. Con las etiquetas *waste* (residuos) y *litter* (basura) Google indica la presencia de basura en la imagen, en cambio IBM no muestra ninguna etiqueta con ese significado.

[23] Elementos que se encuentran en las fotos, como pueden ser autos, personas, calle, basura, etc.

Figura 5.9: Tweets sobre limpieza con imágenes

Figura 5.10: Tweet sobre limpieza con imágenes sin palabras clave

Figura 5.11: Respuesta de los servicios provistos por Google (izquierda) y por IBM (derecha)

Figura 5.12: Respuesta de los servicios provistos por Google (izquierda) y por IBM (derecha) para una foto sin basura

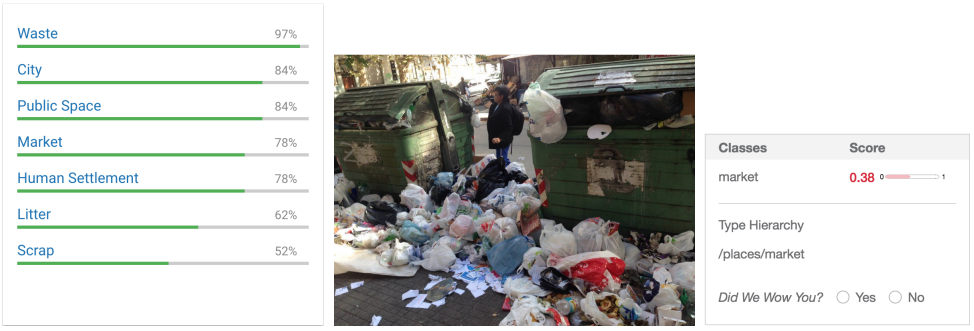


Figura 5.13: Respuesta de los servicios provistos por Google (izquierda) y por IBM (derecha) para una foto con basura

Tomando en cuenta la calidad de los resultados para uno y otro servicio, se optó por utilizar el servicio provisto por Google.

Concretamente, el Módulo de procesamiento de imágenes implementado añade información sobre los tweets recolectados que tengan adjunta alguna imagen. Usando su URL, se obtiene un *array de bytes* que representa la imagen y se realiza una petición a la Cloud Vision API de Google. Una vez obtenida la respuesta, se almacenan las diferentes etiquetas detectadas como información extra a la Pieza de Información.

Si bien los resultados obtenidos utilizando la herramienta Cloud Vision API de Google como se muestra en la Sección 6.3 son muy buenos, posee la limitante de que, a partir de una cierta cantidad de peticiones, deja de ser gratuita y que depende de una conexión a Internet. Como forma de contrarrestar estas limitantes, se decidió a la hora de obtener los tweets, tomar en cuenta únicamente los que ya fueron procesados por el Módulo de georreferenciación, ya que en este módulo se intentan descartar tweets que no pertenecen a Montevideo. También, se decidió que, en caso de fallar la conexión con la API, se realice un reintento de la petición.

5.3.5.2. Construcción corpus de imágenes

El primer desafío para poder entrenar un modelo de reconocimiento de imágenes era tener un *corpus* de imágenes lo suficientemente grande para lograrlo. Este *corpus* debía tener imágenes de dos clases:

- Imágenes que contengan contenedores de Montevideo.
- Imágenes que no contengan contenedores de Montevideo.

Para conseguir las imágenes que contengan contenedores, en la Figura 5.14 se puede ver algunos ejemplos, se utilizaron tres métodos,:

Figura 5.14: Imágenes de Montevideo que contienen contenedores

1. Tomar fotografías de los contenedores de la ciudad.
2. Guardar las imágenes adjuntas a los tweets recolectados.
3. Realizar búsquedas de imágenes en Google.

Para conseguir las imágenes de Montevideo sin contenedores se procedió a realizar búsquedas en Google. En la Figura 5.15 se pueden ver algunos ejemplos.

Figura 5.15: Imágenes de Montevideo que no contienen contenedores

De esta forma, entre ambas categorías, se obtuvieron aproximadamente 450 imágenes, 200 de ellas contenían contenedores. Esta cantidad de imágenes resultaba demasiado pequeña, por lo que se buscó una alternativa para aumentarla.

La alternativa encontrada fue utilizar un método provisto por la librería de Python Keras, en el que, aplicando diferentes transformaciones, se crean imágenes nuevas a partir de las ya existentes, como se muestra en Figura 5.16.

Figura 5.16: Imagen generada por el método. A la izquierda la imagen original

Utilizando este método se logró conseguir un *corpus* de, aproximadamente, 4000 imágenes con el que se procedió a entrenar el modelo.

5.3.5.3. Generalización

Al utilizar la Cloud Vision API de Google permite que para la generalización de ciudad no haya que realizar ninguna modificación y que la generalización del dominio sea muy sencilla, ya que únicamente consiste en modificar las etiquetas que son tomadas en cuenta para el dominio.

En caso de realizar el entrenamiento de un modelo específico para la ciudad y dominio, para su generalización será necesario construir un *corpus* de imágenes correspondientes a la ciudad y dominio elegidos y volver a entrenar el modelo.

5.3.6. Módulo clasificador de dominio: reclamos

El objetivo de este módulo es determinar en la Etapa 2 si un tweet pertenece al conjunto de los tweets en español que pueden considerarse reclamos o denuncias. En esta sección se describe el *corpus* utilizado, el modelo construido y el ajuste paramétrico realizado. Los resultados se explican en detalle en la Sección 6.4.

5.3.6.1. Construcción del corpus

Para la construcción del *corpus* se utiliza un subconjunto de la colección de reclamos del Sistema Único de Reclamos de la IM (SUR). Los reclamos fueron cedidos por el organismo a través de un pedido de información. Están clasificados por fecha y categoría. Para su utilización en este trabajo se almacenan en una base de datos MongoDB y previo a su importación, cada uno de los archivos pasa por un proceso de depuración de los datos utilizando Pentaho Data Integration. Durante el proceso de depuración se eliminan columnas vacías y se cambia la extensión de cada uno de los archivos a CSV, tipo de archivo que permite almacenar datos con estructura de tabla. Para su importación se utiliza la librería de Python Pandas. Debido a la gran cantidad de reclamos y el tiempo requerido para su importación se decidió utilizar únicamente los correspondientes a los años 2015 y 2016.

Dada la falta de disponibilidad de tweets sobre reclamos de limpieza, el conjunto de reclamos ofrece la mejor aproximación disponible de datos que en su contenido son reclamos de ciudadanos. Si bien el medio por el que estos reclamos fueron realizados no es Twitter, su longitud permite suponer una similaridad aceptable.

Adicionalmente se utilizó el *corpus* Tass General 2015, que cuenta con aproximadamente 60 mil tweets en español escritos por diferentes personalidades influyentes de diversos ámbitos y países. La temática de los tweets abarca política, fútbol, literatura y entretenimiento y permite suponer la ausencia de reclamos en su contenido.

Se agregaron también unos 8 mil tweets que fueron descartados previamente por el Módulo de georreferenciación por ser originados en otros países, son de especial interés porque la mayoría no son en español.

Se seleccionaron de forma aleatoria 68 mil reclamos de un total de 121 mil de forma de obtener un *corpus* compuesto por 50 % de documentos de cada clase: Reclamos y No-Reclamos.

El 33 % de los documentos del *corpus* fue separado como conjunto de test y no fue utilizado en ninguna etapa del entrenamiento del clasificador.

5.3.6.2. Descripción del clasificador

En el contexto de un modelo de aprendizaje supervisado, el objetivo de un clasificador es, a partir de una colección de documentos etiquetados, poder predecir la etiqueta de un documento nunca visto antes. En este módulo cada documento está etiquetado como Reclamo o como No-Reclamo.

Para la construcción del clasificador se entrenó un estimador basado en el algoritmo SVM usando la implementación LinearSVC[24], de la librería de Python Scikit-Learn.

LinearSVC fue el estimador que mejor desempeño tuvo en pruebas preeliminares entre resultados y tiempos de ejecución.

Cada documento fue preprocesado de manera de eliminar símbolos extraños, URLs y espacios en blanco. En todas las etapas de entrenamiento y test se utiliza la versión preprocesada de los documentos.

La librería Scikit-Learn proveé una serie de transformadores que convierten una colección de documentos en cierta representación matemática a ser usada posteriormente. Para este clasificador se utilizaron los transformadores de Scikit-Learn CountVectorizer y TfIdfTransformer y uno desarrollado especialmente para este módulo: SpanishLanguageDetector.

- CountVectorizer convierte los documentos en una matriz de conteo de palabras, cada fila es un documento y cada columna es una de las palabras que aparecen en la colección.
- TfIdfTransformer convierte una matriz de conteo de palabras en una matriz normalizada utilizando la representación Tf-Idf, que busca disminuir el impacto de palabras que ocurren muy frecuentemente en el *corpus* y que por lo tanto son poco informativas.
- SpanishLanguageDetector es un transformador implementado para este trabajo, basado en dos liberías de detección de idiomas LangId y LangDetect. El transformador convierte una colección de documentos en una matriz unidimensional donde cada fila representa un documento y la columna representa con un 1 si se detectó español y con un 0 si se detectó otro idioma.

El clasificador es una composición de varios pasos entre los transformadores y el estimador LinearSVC. La concatenación de las matrices resultantes de SpanishLanguageDetector y la combinación de CountVectorizer y TfIdfTransformer son la base sobre la que el estimador aprenderá para poder predecir la clase de nuevos documentos.

5.3.6.3. Ajuste parámetrico

Cada transformador de Scikit-Learn así como los estimadores tienen una serie de parámetros cuya variación puede resultar en un modelo mejor ajustado al problema particular.

Para buscar el mejor desempeño del clasificador es necesario entrenar y evaluar el modelo completo con cada combinación de parámetros.

GridSearchCV es una herramienta de Scikit-Learn que permite realizar el ajuste sobre distintos parámetros eligiendo la mejor combinación al finalizar el proceso.

De acuerdo a la documentación de Scikit-Learn se decidieron explorar los parámetros ngram_range de CountVectorizer y C de LinearSVC.

[24] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

- El parámetro ngram_range especifica si el conteo se hará sobre unigramas (una palabra), bigramas (dos palabras), trigramas (tres palabras) o una combinación de estos. Este tipo de conteo puede ser interesante si existen expresiones de más de una palabra que se repiten en el *corpus*.

- En SVM se buscan dos cosas: por un lado un hiperplano cuya distancia mínima a los ejemplos sea lo más grande posible y por otro un hiperplano que separe los ejemplos de clases distintas lo mejor posible. El parámetro C determina en qué medida se favorece un objetivo sobre otro. Con un valor bajo de C es posible que haya más errores en la etapa de entrenamiento pero el modelo generalice más correctamente en la etapa de test. Con un valor más alto de C habrá menos errores en la etapa de entrenamiento comprometiendo la capacidad de generalización del modelo.

GridSearchCV realiza una evaluación utilizando *cross validation* para cada combinación de parámetros. La comparación de resultados se realiza utilizando indicadores como *precision* o *recall*.

5.3.6.4. Generalización

El módulo es utilizable por la plataforma independientemente de la ciudad elegida, en la medida que exista un *corpus* de reclamos con los que trabajar. Con respecto al dominio, si es referente a reclamos en general este módulo no debería sufrir cambios.

5.3.7. Módulo clasificador de dominio: limpieza

El objetivo de este módulo es determinar durante la Etapa 2 si un tweet pertenece al conjunto de los tweets que tratan sobre limpieza. En esta sección se describe el *corpus* utilizado, el modelo construido y el ajuste paramétrico realizado. Los resultados se explican en detalle en la Sección 6.5.

5.3.7.1. Construcción del corpus

En este módulo se utilizan los reclamos cedidos por la IM, como se explicó en 5.3.6.1. Los reclamos realizados a través del SUR ofrecen una buena aproximación a reclamos hechos a través de Twitter. Al estar categorizados por área es posible separar los reclamos que tratan sobre limpieza de los que no.

Para determinar a qué clase pertenece cada reclamo, se utilizó un campo que indica a qué área (limpieza, tránsito, etc.) pertenece un reclamo. De esta forma, se obtiene un *corpus* con aproximadamente 50 % de los reclamos pertenecientes a la clase Limpieza y 50 % reclamos de la clase No-Limpieza.

5.3.7.2. Descripción del modelo

Como se vio en 5.3.6.2 los clasificadores en aprendizaje automático predicen la clase a la que pertenecen los documentos. Es posible también definir modelos que predigan la probabilidad de un documento de pertenecer a cierta clase. Se dice que estos modelos resuelven un problema de regresión.

Para la construcción de este módulo se hicieron pruebas con modelos de clasificación y modelos de regresión. Una vez entrenado el modelo de regresión es posible elegir un umbral para obtener un clasificador: los documentos se clasificarán dentro de una clase

si la probabilidad está por encima del umbral definido. La elección del umbral se hará de manera de favorecer la *precision* o el *recall*. Se optó por el modelo de regresión ya que da más flexibilidad a la hora de componer resultados con otros módulos como se verá en 5.3.8.

Cada documento fue preprocesado de manera de eliminar símbolos extraños, URLs y espacios en blanco. En todas las etapas de entrenamiento y test se utiliza la versión preprocesada de los documentos.

Al igual que en 5.3.6.2 este modelo es una composición de los transformadores de Scikit-Learn: CountVectorizer y TfIdfTransformer junto a un estimador llamado SVC. SVC es otra implementación de SVM que permite la predicción de probabilidades. La predicción del modelo para un nuevo documento será un valor real entre 0 (No-Limpieza) y 1 (Limpieza).

5.3.7.3. Ajuste paramétrico

Al igual que en 5.3.6.3 se utiliza GridSearchCV para realizar el ajuste de parámetros. Los parámetros elegidos para explorar y los valores de búsqueda son los mismos que para el modelo clasificador de reclamos.

5.3.7.4. Generalización

La implementación de este módulo depende fuertemente del dominio elegido, las decisiones tomadas para la elección del modelo de aprendizaje a entrenar están relacionadas al *corpus* de reclamos. Si el dominio elegido fuera otro presente en el *corpus* de reclamos de la IM, los cambios en este módulo serían mínimos.

5.3.8. Módulo de detección de Eventos

Como se explicó en 5.2.1 en la Etapa 4 la plataforma o bien descarta una Pieza de Información o bien la almacena como Evento.

El objetivo de este módulo es determinar a partir de los resultados de los módulos de la Etapa 2 cuándo una Pieza de Información es un Evento.

5.3.8.1. Construcción del corpus

El *corpus* consiste en los tweets obtenidos por el módulo de extracción de tweets en el período reportado en 5.3.1. Cada uno de esos tweets se anotó manualmente con una de cuatro clases: "MUY_UTIL", "UTIL", "POCO_UTIL" y "NADA_UTIL" con los criterios y procedimiento que se explican en 5.3.2. A los efectos de este módulo se considerará un tweet etiquetado con "MUY_UTIL" como un Evento y a los demás como No Eventos.

5.3.8.2. Árbol de decisión

El enfoque inicial fue construir un árbol de decisión que usando las salidas de los módulos de la Etapa 3 como *features* pudiera clasificar una Pieza de Información como Evento o No Evento.

Como se explica en 2.2.4.1 los árboles de decisión son un modelo simple que mediante lógica booleana permiten obtener una clasificación rápida para cada ejemplo al tiempo que permite visualizar la importancia relativa de las *features* involucradas.

En el caso particular donde las *features* son el resultado de los módulos de la Etapa 2, un árbol de decisión permite analizar con claridad cómo se relacionan estos módulos y qué importancia tienen en la decisión final.

El problema con el uso de este enfoque es que los árboles de decisión se caracterizan por sobreajustar el modelo de aprendizaje al conjunto de entrenamiento perdiendo capacidad de generalizar predicciones.

Sumado a lo anterior, el conjunto de entrenamiento para este problema es pequeño por lo que el sobreajuste es un problema mayor.

5.3.8.3. Random Forests

Random Forests [31] es un método de ensamblado de árboles de decisión que a partir de la construcción de varios árboles independientes durante el entrenamiento, predice la clase de un nuevo ejemplo tomando la clase predicha por la mayoría de los árboles. Este método tiene la ventaja de minimizar el efecto de sobreajuste de los árboles individuales.

El modelo construido es una composición de cuatro transformadores, uno por módulo de la Etapa 2, y un estimador RandomForestClassifier de la librería Scikit-Learn. Cada transformador toma una Pieza de Información y devuelve un valor que corresponde a la salida de cada módulo. Los cuatro valores conforman el vector de *features* que será clasificado por el modelo como Evento o No Evento.

5.3.8.4. Ajuste paramétrico

Al igual que en los clasificadores de los otros módulos, se utiliza GridSearchCV de Scikit-Learn para elegir la mejor combinación de parámetros buscando mejorar el *recall*. De acuerdo a la documentación de RandomForestClassifier se decidieron explorar los parámetros "criterion", "class_weight" y "n_estimators".

- El parámetro "criterion" define qué función se utilizará para decidir qué feature divide mejor al árbol. Las opciones son "gini" o "entropy".
- El parámetro "class_weight' define el peso relativo que tiene cada una de las clases. La opción "balanced" determina un peso general inversamente proporcional a la frecuencia de los ejemplos para todos los árboles, "balanced_subsample" define pesos de igual manera pero calculados para cada árbol.
- El parámetro "n_estimators" define cuántos árboles serán generados, las opciones son 10,15 y 20.

En la Figura 5.17 se puede ver uno de los árboles generados por el modelo, y en la Sección 6.6 se reportan los resultados para este módulo.

5.3.8.5. Generalización

El modelo de clasificación deberá ser redefinido conforme cambien las salidas de los módulos de la Etapa 2. Sin embargo el resto de los conceptos de este clasificador son independientes de la ciudad y el dominio, haciéndolo fácilmente generalizable a otros.

5.3.9. Visualización de los Eventos

El objetivo de este módulo es la visualización por parte del usuario de los Eventos detectados por el sistema.

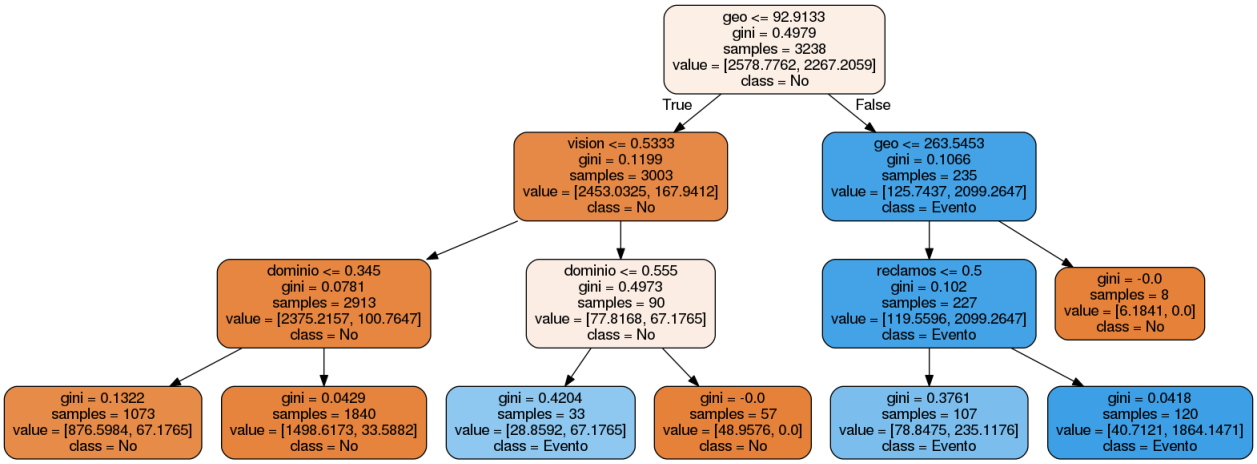


Figura 5.17: Uno de los árboles generados por RandomForestClassifier

5.3.9.1. Descripción

Para la visualización de los Eventos detectados por el Módulo de detección de eventos, explicado en 5.3.8, se decidió desarrollar una página web que se encuentra alojada en `http://krypton.mgcoders.uy`.

Para la implementación de la web se utilizó AngularJS y NodeJS.

La página principal de la web, que se puede observar en la Figura 5.1, muestra un mapa centrado en el departamento de Montevideo. Para la visualizacón del mapa se utiliza Leaflet. Esta herramienta permite mostrar información de diferentes fuentes, que son separadas en capas. Para la visualización de los Eventos se utilizaron tres capas:

Mapa

Es la capa que dibuja el mapa, obtenido desde OSM. El mapa mostrado se encuentra centrado para la ciudad de Montevideo y posee tanto las calles como los lugares de interés de la ciudad.

Esta capa se puede observar en la Figura 5.20a.

Eventos

Capa que muestra los diferentes Eventos encontrados, ubicados en el mapa. Los Eventos pertenecen a un intervalo de tiempo, que está definido por el *slider* que se encuentra en la parte inferior de la web, que se puede ver en la Figura 5.1.

En la Figura 5.20b se puede apreciar esta capa.

La librería Leaflet proveé un mecanismo para visualizar los marcadores cercanos agrupados. Como se puede ver en la Figura 5.20b a medida que aumenta la cantidad de marcadores agrupados, cambia el color de la agrupación.

Evento seleccionado

Una vez seleccionado uno de los Eventos mostrados por la capa de Eventos, se hace visible esta capa que muestra la información de un Evento en particular. Lo que se muestra de cada Evento son las intersecciones encontradas por el Módulo de georreferenciación junto con los elementos del dominio cercanos.

Además de mostrar la información de las intersecciones y los contenedores en el mapa, se abre un cuadro en el que se muestra información del tweet que generó el Evento. En la Figura 5.18 se puede ver el cuadro para un Evento. En la parte inferior del cuadro se pueden observar cuatro botones:

Figura 5.18: Cuadro que muestra información del tweet que generó el evento

- Cerrar: se utiliza para deseleccionar el Evento y visualizar nuevamente la capa de Eventos.
- Resolver: corresponde a una de las funcionalidades explicadas en 7.2.2, que permite resolver un Evento e informarle a las personas que hayan realizado el reclamo.
- Asignar: corresponde, también, a una de las funcionalidades mencionadas en 7.2.2. Esta funcionalidad permite asignar un Evento a una cuadrilla de limpieza.
- No es evento: permite eliminar de la colección de Eventos a los falsos positivos generados por el sistema. Esta funcionalidad forma parte de las explicadas en 7.2.2.

Como muestra la Figura 5.19 si el tweet asociado con el Evento seleccionado contiene adjunta una imagen, es mostrada presionando sobre la miniatura que se encuentra en el cuadro.

La información mostrada tanto por la capa de Eventos como por la capa de Evento seleccionado se obtiene a través de un servidor web, que publica una API rest, implementado utilizando Python y Flask, de igual forma que el explicado en 5.3.2.

5.3.9.2. Generalización

La página web para la visualización de los Eventos detectados por el sistema se desarrolló pensando en poder modificar el dominio y que no sea necesario ningún tipo de modificación del código. Al estar los datos almacenados en una base de datos y consumirlos mediante una API rest, el dominio no afecta a la visualización. Por otra parte, para modificar la ciudad lo único que es necesario realizar es cambiar las coordenadas en las que el mapa se encuentra centrado al inicio, siendo esto muy sencillo de hacer.

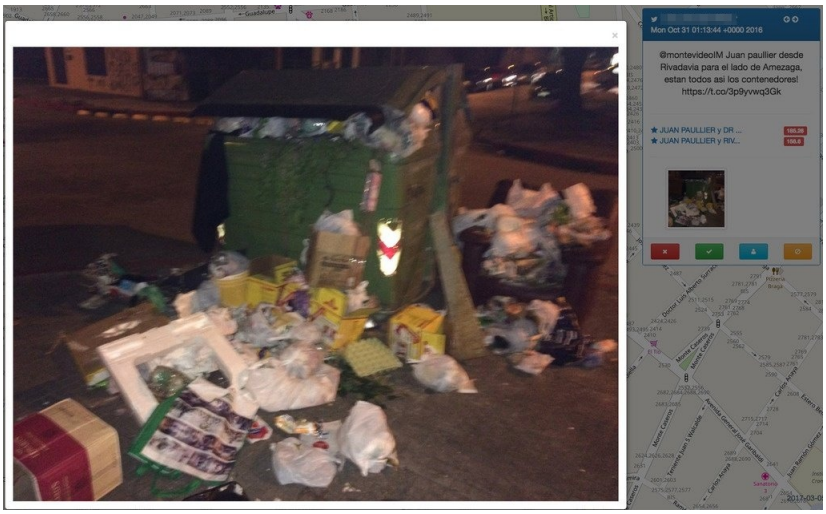
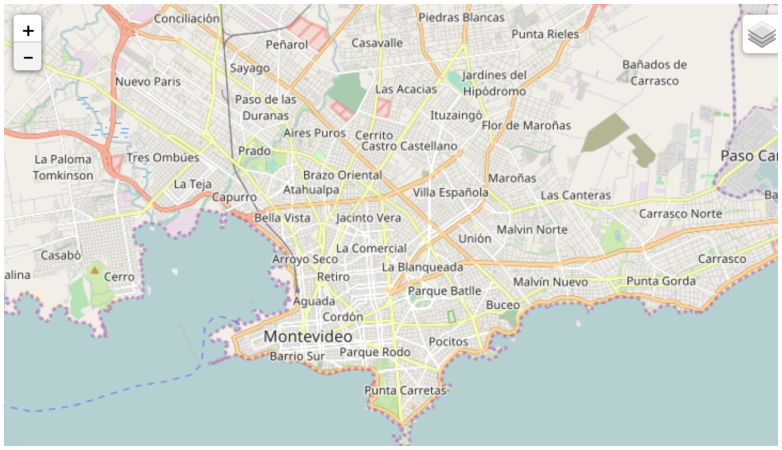
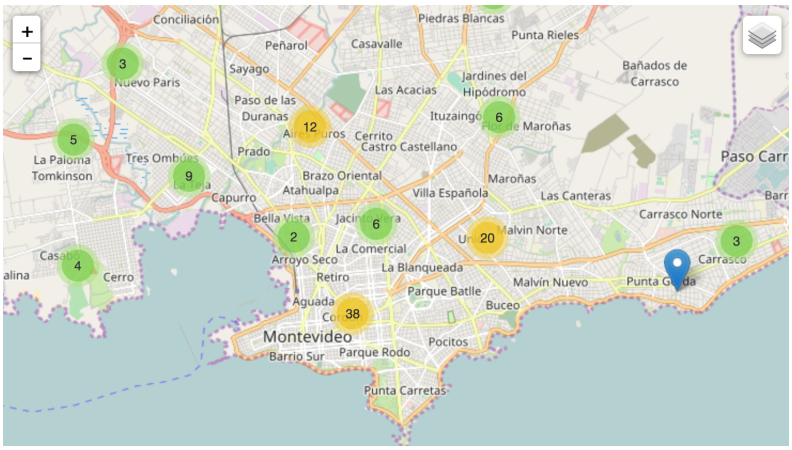


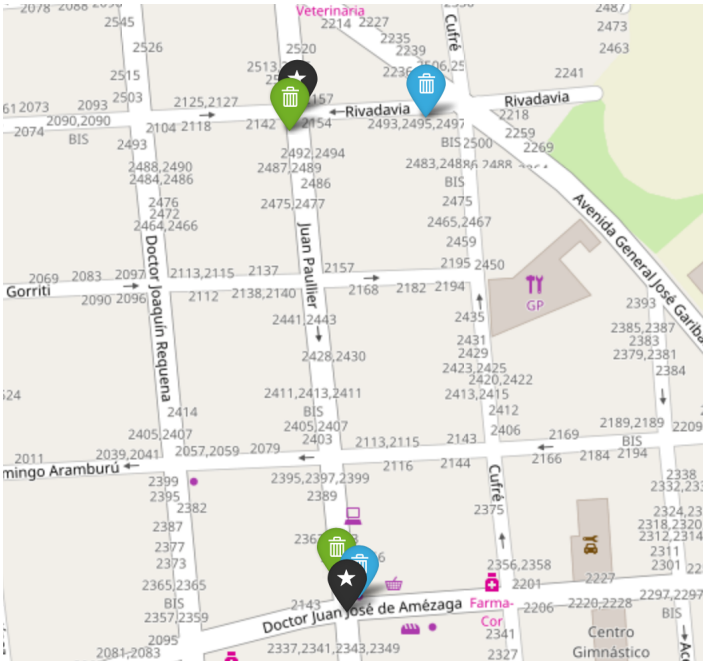
Figura 5.19: Visualización de la imagen del tweet que generó el Evento seleccionado



(a) Capa de mapa, obtenida de OSM



(b) Capa de Eventos



(c) Capa de Evento seleccionado para el ejemplo de Figura 6.4

Figura 5.20: Capas

Capítulo 6

Resultados obtenidos

En este capítulo se presentan los resultados obtenidos por los diferentes módulos de la plataforma, y ejemplos que ilustran su funcionamiento.

6.1. Módulo de extracción de tweets

Desde la fecha 27-09-2016 hasta la fecha 10-01-2017 el módulo recuperó 15528 tweets, que después de ser filtrados por el Módulo de georreferenciación como se explica en 5.3.4, y de ser categorizados por el módulo de anotación, como se explica en 5.3.2, se dividen como se observa en la Tabla 6.1.

	Cantidad
Descartados por Georreferenciación	7857
Etiquetados como MUY_UTIL	238
Etiquetados como UTIL	133
Etiquetados como POCO_UTIL	2046
Etiquetados como NADA_UTIL	5254
Total	15528

Tabla 6.1: Tweets importados

Para los tweets descargados, el promedio de demora desde que un tweet es creado hasta que es importado por el módulo es de 0.64 segundos; el tiempo máximo es de 484 segundos y la desviación estándar es de 7.16 segundos (Figura 6.1).

Se puede ver en la Figura 6.2 la distribución de descarga de tweets no descartados en el tiempo. Si bien la cantidad de los datos no permite hacer un análisis completo, se pueden relacionar los picos a ciertos momentos de crisis en la recolección de la basura, el caso claro es la segunda quincena de Diciembre. Si bien no es el objeto principal de este trabajo, esta correlación puede ser útil para el análisis de posicionamiento en medios de comunicación.

6.2. Módulo de georreferenciación

Del total de tweets importados, el Módulo de georreferenciación descartó 7857 por tratarse de tweets georreferenciados fuera de la ciudad de Montevideo.

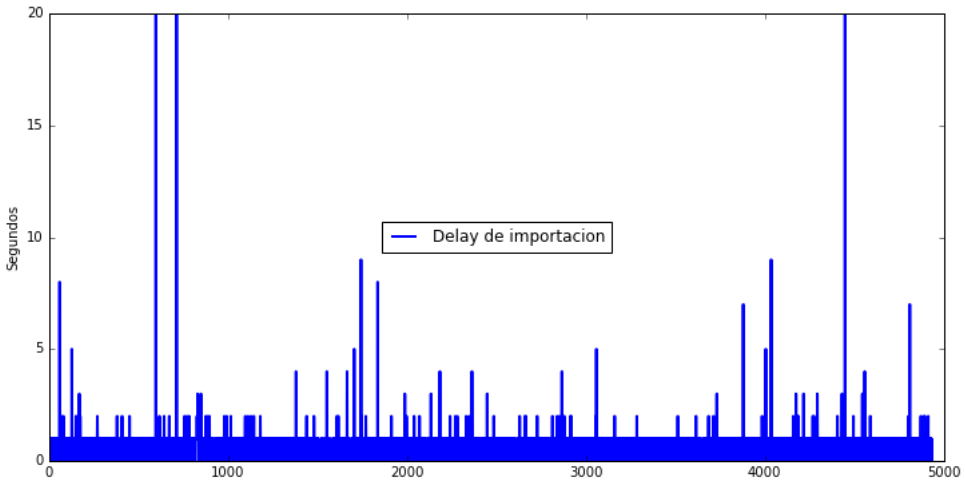


Figura 6.1: Demoras de importación

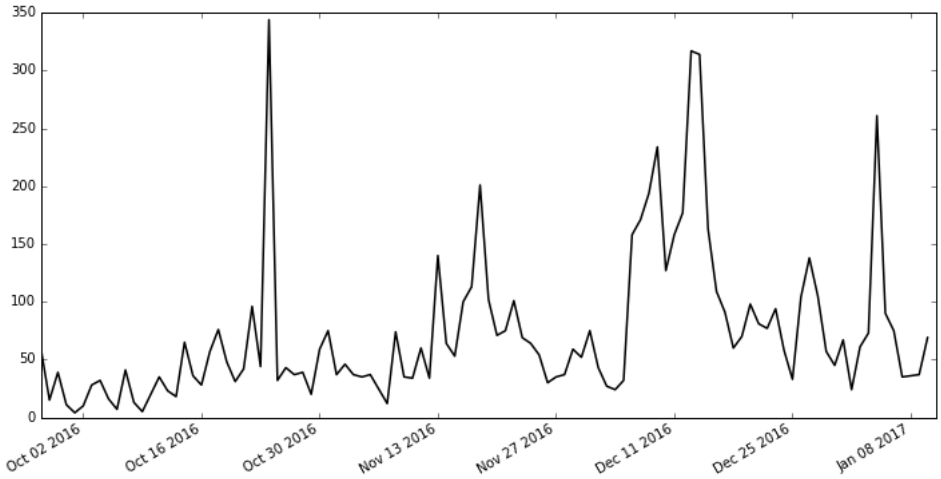


Figura 6.2: Cantidades de Tweets importados según fecha

De los 238 tweets anotados con la etiqueta MUY_UTIL como se explica en 5.3.2, el Módulo de georreferenciación encontró soluciones correctas para 213 de ellos. De los 25 restantes, 12 no tuvieron ninguna solución y 13 tuvieron soluciones equivocadas.

Las razones por las que el algoritmo falla para estos 25 casos son las siguientes:

- Errores de ortografía en el nombre de calles o lugares: el algoritmo realiza la búsqueda de texto exacto.
- Direcciones con numeración: el algoritmo no prevé búsquedas con numeración, solo intersecciones entre calles y entre calles y lugares.
- Las referencias geográficas no intersectan en un radio de 150m que es lo definido por el algoritmo.
- Las referencias geográficas corresponden a nombres de calles que cambiaron.
- La solución correcta no se encuentra entre las cuatro más puntuadas, y el algoritmo la descarta.

En la Figura 6.3 se pueden ver ejemplos; "Misissipi" mal escrito, "8 de Octubre 4630" dirección con numeración y "propios" una calle que ya no existe.

Figura 6.3: Ejemplos donde el algoritmo falla

Al construir el algoritmo de este módulo se tuvo que lograr un balance entre exhaustividad y performance, sin tomar en cuenta aquellos casos de situaciones para las cuales había muy pocos ejemplos. Aún al finalizar el período de recolección de datos para reportar los resultados de este trabajo, la cantidad de ejemplos para los que el Módulo de georreferenciación falla es bajo. No obstante lo anterior, en 7.2.1 se discuten alternativas

y mejoras a ciertos aspectos del algoritmo para poder atender algunas de las fallas más comunes.

En la Figura 6.4 se puede ver un caso de la situación más frecuente en cuanto a las soluciones encontradas. Las dos soluciones corresponden a intersecciones presentes en el texto. Ambas tienen puntaje alto y la diferencia entre puntajes se debe a que el algoritmo encontró contenedores más cercanos a una que a otra.


```
{
    "info_base": "JUAN PAULLIER",
    "puntaje": 185.25925352208787,
    "geohash": "6cb19c92",
    "ubicacion": {...},
    "interseccion_encontrada": "JUAN PAULLIER y DR JUAN J DE AMEZAGA"
},
{
    "info_base": "JUAN PAULLIER",
    "puntaje": 156.5986550126261,
    "geohash": "6cb19f15",
    "ubicacion": {...},
    "interseccion_encontrada": "JUAN PAULLIER y RIVADAVIA"
}
```


Figura 6.4: Ejemplo de intersección de calles

Es interesante notar algunos de los ejemplos donde el texto no hace referencia al tradicional cruce de calles y sin embargo el algoritmo logra la identificación de una solución.

Se ve en la Figura 6.5 que la solución más puntuada es la correcta. El algoritmo logró identificar la intersección de un lugar (Escuela Sanguinetti) y una calle (Munar) en un radio de 150m.

En el ejemplo de la Figura 6.6, la solución correcta es la cuarta mejor puntuada; la palabra "Médica" al utilizar un índice con *stemming* en la base de datos devolvió resultados para "Médica" y "Médico" así como "Uruguaya" devolvió resultados para "Uruguay". Es interesante notar también que la primera y la tercera solución son la misma intersección generadas inversamente. En este caso, el *geohash* no coincide y por lo tanto se generaron dos soluciones en lugar de una. Este tipo de ejemplos con soluciones duplicadas se podrían eliminar ajustando el tamaño del *geohash*, es decir la precisión. Sin embargo al hacerlo se podrían perder intersecciones diferentes por estar cercanas entre sí.

Como se verá más adelante (Sección 6.6) el resultado de este módulo sobre una Pieza de Información es determinante en su clasificación como Evento. Más aún se verá que


```
{
    "info_base": "ESCUELA 76 GRUPO ESCOLAR F SANGUINETTI",
    "puntaje": 126.4403934633949,
    "geohash": "6cb1d6sg",
    "ubicacion": { ... },
    "interseccion_encontrada": "MARIA STAGNERO DE MUNAR",
}
```

Figura 6.5: Ejemplo de intersección de calle y lugar

las Piezas de Información que no se logran clasificar correctamente como Eventos son en general las mismas que presentan errores en este módulo.

6.3. Módulo de procesamiento de imágenes

Como se explicó en 5.3.5 para la construcción de este módulo se utilizó la Cloud Vision API provista por Google. Como se puede ver en la Tabla 6.2, más allá de contar con un *corpus* pequeño de Piezas de Información con imágenes adjuntas, los resultados obtenidos por este módulo son favorables.

	precision	**recall**	**f1-score**	**ejemplos**	**% del corpus**
Sin-Basura	0.86	1.00	0.93	840	66.5 %
Con-Basura	1.00	0.68	0.81	423	33.5 %
Total	0.91	0.89	0.89	1263	100 %

Tabla 6.2: Resultados sobre tweets con imágenes

Para obtener los resultados se tomaron en cuenta los tweets recolectados que tenían alguna imagen adjunta. Esos tweets fueron recorridos manualmente para determinar cuántos habían sido clasificados incorrectamente por el módulo. La matriz de confusión obtenida se muestra en Tabla 6.3.

En la Figura 6.7 se puede apreciar un ejemplo de un falso negativo junto con las etiquetas devueltas por la Cloud Vision API provista por Google.

Al finalizar este trabajo, y como se puede ver en los modelos generados en la Sección 6.6 para clasificar una Pieza de Información como Evento surge que la influencia de este módulo en esta clasificación es baja; en general las imágenes ilustran una realidad expresada en el texto ocasionando cierta redundancia entre módulos. Esta redundancia es más notable cuando se configura el Módulo de extracción de tweets para funcionar con palabras clave relacionadas al dominio. En una situación diferente, por ejemplo haciendo


```
{
    "info_base": "SERVICIO MEDICO INTEGRAL - SANATORIO ITALIANO",
    "puntaje": 139.34921602612746,
    "geohash": "6cb13zmf",
    "ubicacion": {...},
    "interseccion_encontrada": "AV 8 DE OCTUBRE"
},
{
    "info_base": "BIBLIOTECA CENTRAL DE SERVICIOS MEDICOS - BSE",
    "puntaje": 137.93540211520812,
    "geohash": "6cb13q01",
    "ubicacion": {...},
    "interseccion_encontrada": "AV URUGUAY",
},
{
    "info_base": "AV 8 DE OCTUBRE",
    "puntaje": 123.57968442808043,
    "geohash": "6cb13zm9",
    "ubicacion": {...},
    "lugar_encontrado": {
        "geometry": {...},
        "nombre": "SERVICIO MEDICO INTEGRAL - SANATORIO ITALIANO"
    }
},
{
    "info_base": "MEDICA URUGUAYA - SANATORIO",
    "puntaje": 122.35257313544922,
    "geohash": "6cb19bp9",
    "ubicacion": {...},
    "interseccion_encontrada": "AV 8 DE OCTUBRE",
}
```


Figura 6.6: Ejemplo de varias soluciones

		Clase Estimada	
		Sin-Basura	Con-Basura
Clase Real	Sin-Basura	840	0
	Con-Basura	135	288

Falsos Positivos

Falsos Negativos

Tabla 6.3: Matriz de confusión sobre tweets con imágenes

una extracción de todos los tweets en español o todos los relacionados a Montevideo, el Módulo de procesamiento de imágenes cobraría otra relevancia en el resultado final.

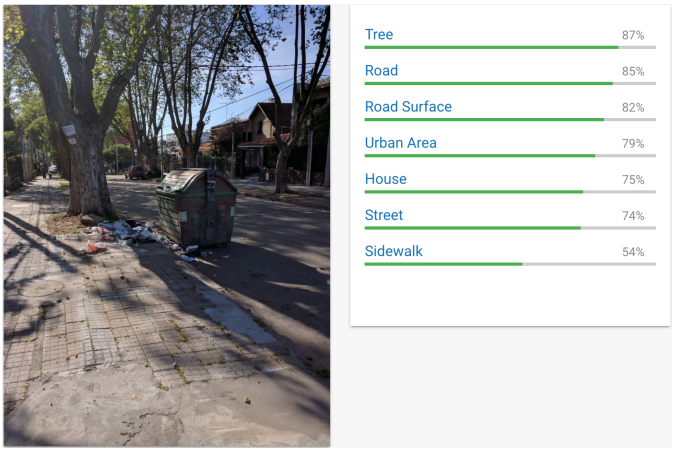


Figura 6.7: Falso Negativo junto con las etiquetas devueltas por la API

6.4. Módulo clasificador de dominio: reclamos

El ajuste paramétrico se hizo sobre los valores que se muestran en la Tabla 6.4, favoreciendo el *recall* mediante una evaluación de *cross validation* con cinco particiones.

En un notebook Lenovo con CPU: Intel(R) Core(TM) i5-3320M, @ 2.60GHz y 16 GB de RAM el ajuste paramétrico y el entrenamiento sobre el conjunto de entrenamiento (86620 ejemplos) tardó 48 minutos.

Los mejores parámetros resultaron ser C=1, parámetro que como se explicó en 5.3.6.3 determina en qué medida se favorece un objetivo sobre otro, y ngram_range = unigrama.

Parámetro	**Valores**	**Valor elegido**
ngram_range	unigramas,bigramas,trigramas	unigrama
C	0.1, 1, 10	1

Tabla 6.4: Parámetros

En la Tabla 6.5 se muestran varios indicadores de la evaluación del clasificador sobre el conjunto de test (42664 ejemplos) y en la Tabla 6.6 se muestra la matriz de confusión.

Los resultados sobre el conjunto de test son excepcionalmente buenos, sin embargo es bueno recordar que el modelo será utilizado sobre tweets y no sobre reclamos del sistema SUR. En Tabla 6.7 se ven los indicadores para el total de tweets no descartados y en Tabla 6.8 la matriz de confusión. En este caso si bien el *recall* es de un aceptable 84 % la *precision* es de solo 14 % debido a la gran cantidad de falsos positivos.

	precision	recall	f1-score	ejemplos	% del corpus
No-Reclamo	1.00	1.00	1.00	22697	17.5 %
Reclamo	1.00	1.00	1.00	19967	14.4 %
Total	1.00	1.00	1.00	42664	33 %

Tabla 6.5: Resultados sobre conjunto de test

		Clase Estimada	
		No-Reclamo	Reclamo
Clase Real	No-Reclamo	22636	61
	Reclamo	69	19898

Falsos Positivos

Falsos Negativos

Tabla 6.6: Matriz de confusión para conjunto de test

	precision	recall	f1-score	ejemplos	% del corpus
No-Reclamo	0.99	0.83	0.91	7433	97.53 %
Reclamo	0.14	0.84	0.24	238	3.12 %
Total	0.97	0.83	0.88	7621	100 %

Tabla 6.7: Resultados sobre el total de tweets

		Clase Estimada	
		No-Reclamo	Reclamo
Clase Real	No-Reclamo	6183	1250
	Reclamo	38	200

Falsos Positivos

Falsos Negativos

Tabla 6.8: Matriz de confusión para el total de tweets

6.5. Módulo clasificador de dominio: limpieza

El ajuste paramétrico se hizo sobre los valores que se muestran en la Tabla 6.9 favoreciendo el *recall* mediante una evaluación de *cross validation* con cinco particiones.

En un notebook Lenovo con CPU: Intel(R) Core(TM) i5-3320M, @ 2.60GHz y 16 GB de RAM el ajuste paramétrico y el entrenamiento sobre el conjunto de entrenamiento (95576 ejemplos) tardó aproximadamente 19 horas.

Los mejores parámetros resultaron ser C=1 y ngram_range = unigrama.

Parámetro	**Valores**	**Valor elegido**
ngram_range	unigramas,bigramas,trigramas	unigrama
C	0.1, 1, 10	1

Tabla 6.9: Parámetros

Como se explicó en 5.3.7.2, para la construcción de este módulo se optó por un modelo de regresión que para cada documento asigna probabilidades de pertenencia a la clase Limpieza. Este tipo de modelos permite variar el umbral por encima del que un documento será considerado dentro de una clase determinada. En la Figura 6.8 se ve el desempeño del clasificador conforme varía el umbral elegido y en Figura 6.9 se ve otra perspectiva de la misma variación.

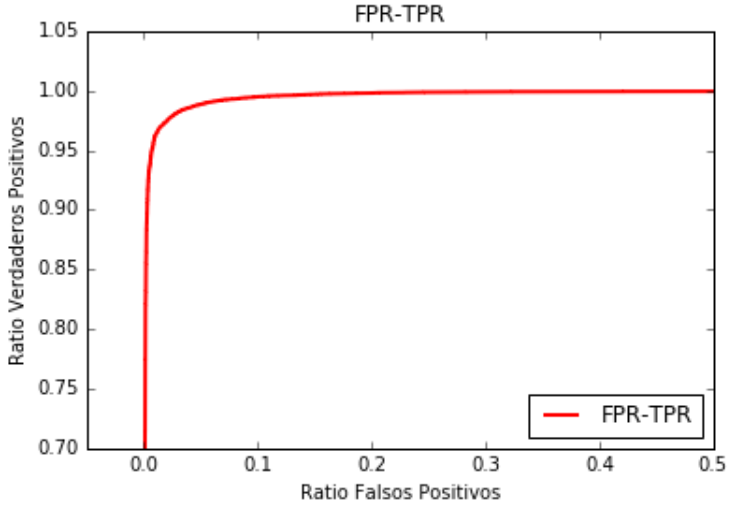


Figura 6.8: Desempeño del clasificador

A modo de ejemplo, tomando el umbral = 0.5 se obtienen los indicadores que se muestran en la Tabla 6.10 evaluando el modelo sobre el conjunto de test (47076 ejemplos). En la Tabla 6.11 se muestra la matriz de confusión para el mismo umbral.

	precision	**recall**	**f1-score**	**ejemplos**	**% del corpus**
No-Reclamo	0.97	0.98	0.98	22998	16.8 %
Reclamo	0.98	0.98	0.98	24078	16.1 %
Total	0.98	0.98	0.98	47076	33 %

Tabla 6.10: Resultados sobre conjunto de test, umbral = 0.5

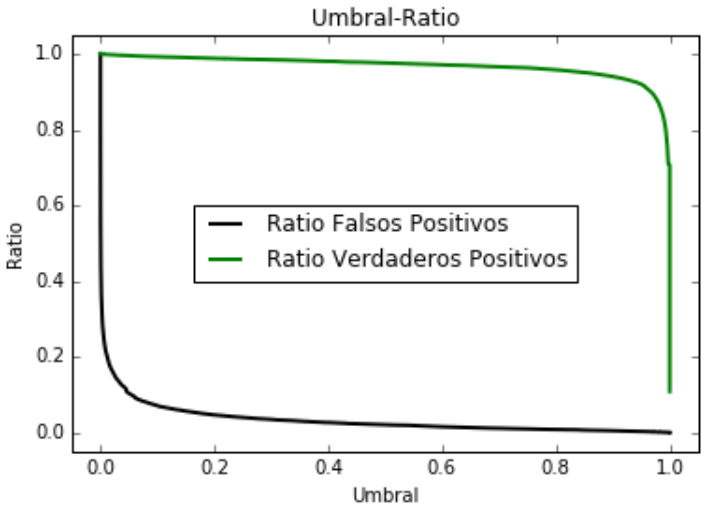


Figura 6.9: Variación del umbral

		Clase Estimada	
		No-Reclamo	Reclamo
Clase Real	No-Reclamo	22508	490
	Reclamo	586	23492

Falsos Positivos

Falsos Negativos

Tabla 6.11: Matriz de confusión para conjunto de test, umbral = 0.5

Como en la Sección 6.4 los resultados sobre el conjunto de test son muy buenos, pero hay que considerar que el modelo será utilizado para clasificar tweets y no reclamos originados en el sistema SUR. Se pueden ver en la Tabla 6.12 los indicadores para la evaluación del modelo sobre el total de tweets y en la Tabla 6.13 la matriz de confusión para el mismo conjunto. De forma similar a la sección anterior se obtiene un buen *recall* para la clase buscada pero una muy mala *precision* causada por el alto número de falsos positivos.

	precision	**recall**	**f1-score**	**ejemplos**	**% del corpus**
No-Reclamo	0.99	0.42	0.59	7433	97.53 %
Reclamo	0.05	0.90	0.09	233	3.12 %
Total	0.96	0.44	0.58	7671	100 %

Tabla 6.12: Resultados sobre el total de tweets, umbral = 0.5

		Clase Estimada	
		No-Reclamo	Reclamo
Clase Real	No-Reclamo	3128	4305
	Reclamo	23	215

Falsos Positivos

Falsos Negativos

Tabla 6.13: Matriz de confusión para el total de tweets, umbral = 0.5

En la Sección 6.6 se mide la importancia de ambos clasificadores de dominio en la cla-

sificación final de una Pieza de Información como Evento. Aún con buen *recall* individual los clasificadores parecen tener poca participación en la clasificación final, especialmente el clasificador de limpieza. Este comportamiento se debe en parte a que los tweets son recuperados mediante palabras clave relacionadas a la limpieza, escenario en el que el módulo tiene poca influencia en el resultado final.

6.6. Resultados de la detección de eventos

El ajuste parámetrico se hizo sobre los valores que se muestran en la Tabla 6.14 favoreciendo el *recall* mediante una evaluación de *cross validation* con cinco particiones.

En un notebook Lenovo con CPU: Intel(R) Core(TM) i5-3320M, @ 2.60GHz y 16 GB de RAM el ajuste paramétrico y el entrenamiento sobre el conjunto de entrenamiento (5139 ejemplos) tardó aproximadamente cuatro minutos.

Los mejores parámetros resultaron ser "criterion"= entropy, "class_weight" = balanced_subsample y "n_estimators" = 15

Parámetro	**Valores**	**Valor elegido**
criterion	gini,entropy	entropy
class_weight	balanced, balanced_subsample	balanced_subsample
n_estimators	10,15,20	15

Tabla 6.14: Parámetros

Como se explicó en 5.3.8.3 las salidas de los módulos de la Etapa 2 actuaron como *features* para el modelo de RandomForests utilizado. Scikit-Learn permite medir la importancia de cada *feature* en el modelo. La importancia es una medida basada en la entropía llamada ganancia de la información o *gain* (2.2.4.1). Se evaluaron las *features* de los árboles del modelo y los valores se ven en la Tabla 6.15

Feature	**Importancia**
Módulo georreferenciación	0.740
Módulo clasificador reclamos	0.175
Módulo clasificador limpieza	0.065
Módulo procesamiento de imágenes	0.018

Tabla 6.15: Parámetros

	precision	**recall**	**f1-score**	**ejemplos**	**% del corpus**
No Evento	1.00	0.96	0.98	2459	97 %
Evento	0.40	0.95	0.56	73	3 %
Total	0.98	0.96	0.97	2532	100 %

Tabla 6.16: Resultados de la detección de Eventos para el conjunto de test

Se pueden ver los resultados para el conjunto de test en la Tabla 6.16 y los resultados y la matriz de confusión asociada para el total del *corpus* en la Tabla 6.17 y la Tabla 6.18. Según estos datos la probabilidad de clasificar un Evento correctamente es de 94 % (*recall*)

	precision	recall	f1-score	ejemplos	% del corpus
No Evento	1.00	0.96	0.98	7433	96.9 %
Evento	0.44	0.94	0.59	238	3.1 %
Total	0.98	0.96	0.97	7671	100 %

Tabla 6.17: Resultados de la detección de Eventos para el total del *corpus*

		Clase Estimada		
		No Evento	Evento	
Clase Real	No Evento	7144	289	*Falsos Positivos*
	Evento	15	223	
		Falsos Negativos		

Tabla 6.18: Matriz de confusión para el total del *corpus*

	Cantidad
Etiquetados como UTIL	19
Etiquetados como POCO_UTIL	115
Etiquetados como NADA_UTIL	134
Total de Falsos Positivos	289

Tabla 6.19: Composición de Falsos Negativos

y la probabilidad de clasificar incorrectamente un Evento, es decir de obtener un falso positivo es de 4 % (*fall-out*).

Es claro que la cantidad de instancias positivas para la etiqueta "Evento" es baja y por lo tanto los resultados deben ser tomados con precaución. Sin embargo según se ve en Figura 6.10 la diferencia de *recall* entre los *corpus* de validación y entrenamiento a partir de cierto número de ejemplos parece estabilizarse lo que indica la capacidad del modelo para generalizar predicciones.

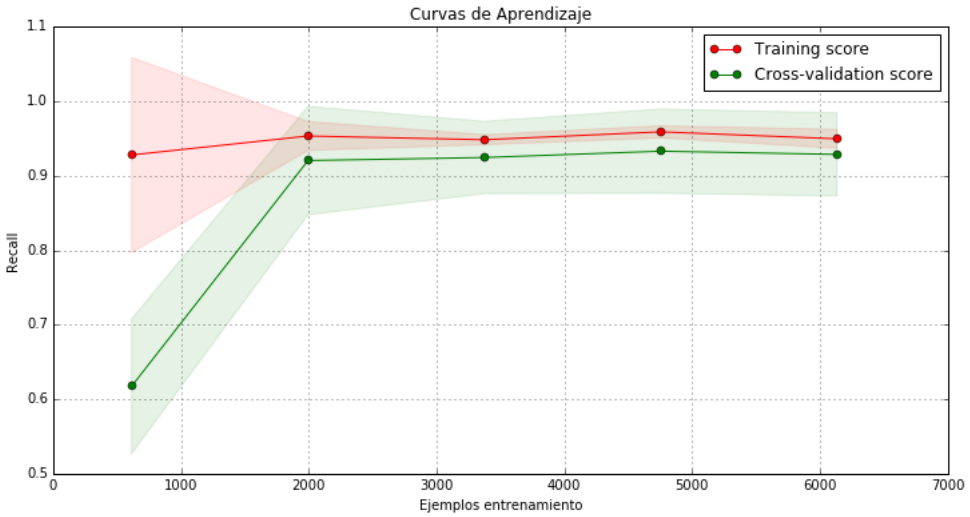


Figura 6.10: Curvas de aprendizaje

Interesa destacar que tanto en los clasificadores de la Etapa 2 como en el clasificador

de Eventos se buscó siempre favorecer el *recall* por sobre la *precision* en la etapa de entrenamiento. La decisión parte del concepto que es preferible tener menor cantidad de falsos negativos independientemente de la cantidad de falsos positivos, esto es: importa más encontrar la mayor cantidad de Eventos posible, aún si esto significa encontrar más cantidad de instancias que no son eventos. En la Tabla 6.19 se ve la composición de los falsos negativos.

Capítulo 7

Conclusiones y trabajo a futuro

7.1. Conclusiones

La revolución tecnológica presentada como motivación para este trabajo es una realidad. De forma explosiva cada vez más ciudadanos interactúan entre sí y con distintas organizaciones día a día. En muchos casos lo hacen sobre la realidad que los rodea con sorprendente cantidad de detalle, acompañando opiniones o constataciones con imágenes y aportando contexto y múltiples puntos de vista sobre una misma realidad. Independientemente de las motivaciones que llevan a estos intercambios; motivos políticos, críticas destructivas o justos reclamos se puede decir que la percepción que se puede obtener sobre lo que sucede en una ciudad es más diversa, integral y exacta en esta nueva era de intercambio de información. Se puede afirmar además que la realidad según se expresa en redes sociales tiene por lo menos en momentos de crisis, cierto grado de correlación con la realidad expresada en medios de comunicación tradicionales.

Hay otro aspecto de la tecnología que hace único el momento en que vivimos: la Inteligencia Artificial (IA), campo del conocimiento en constante desarrollo desde por lo menos 1960 ha dado un gran salto en cuanto a sus aplicaciones prácticas gracias a la abundante cantidad de información generada por los usuarios de Internet. Lo anterior se ha traducido en una carrera entre las grandes empresas, dueñas de los datos, por incorporar la IA a sus negocios empujando aún más el desarrollo de este campo del conocimiento. Como consecuencia práctica, se liberan a la comunidad científica a diario nuevas herramientas y algoritmos que son aplicables a áreas hasta el momento ajenas a estas tecnologías. La industria nacional y los servicios públicos son ejemplos de nuevas áreas donde la IA podría tener un gran impacto.

La generación de datos es la consecuencia del uso masivo de Internet y la principal condicionante para el desarrollo de aplicaciones de IA que sean realmente útiles. Los datos masivos traen aparejados dos aspectos estrechamente vinculados: la preocupación por la privacidad y preservación de datos personales y la búsqueda constante de comercializar ciertos usos de los datos de las personas por parte de las grandes empresas.

En este marco se destaca la política de datos abiertos de ciertas instituciones públicas como la IM, que permite avanzar en la investigación y análisis de estos datos con transparencia y confiabilidad. La disponibilidad de datos abiertos de calidad sobre diferentes aspectos de una ciudad hacen posible trabajos como este, al ofrecer información necesaria y transparente.

En el contexto de este trabajo y durante un período de 10 meses nuestro equipo estudió de cerca la percepción de los ciudadanos sobre la limpieza de la ciudad según la expresan

en Twitter. Se tuvo que estudiar diversas técnicas de extracción y recuperación de datos, mecanismos de almacenamiento, y estrategias para resolver los problemas de obtención de datos confiables a partir de texto en ocasiones ambiguo y escrito informalmente.

Se usaron tecnologías libres, novedosas para el equipo, y técnicas que si bien son accesibles para su uso tienen sólidos fundamentos científicos en diversas disciplinas. Lo anterior sumado a los datos abiertos que ofrece la ciudad permitieron la construcción de un sistema que si bien es mejorable, demuestra la potencialidad que tienen los datos que los ciudadanos generan a diario para la ciudad.

La plataforma construida fue alimentada con 15528 tweets por un espacio de 105 días, estos tweets fueron recuperados mediante palabras clave asociadas a la ciudad de Montevideo y al dominio elegido: la Limpieza. De los 234 tweets manualmente marcados como posibles Eventos, la plataforma identificó el 94 % correctamente mientras obtuvo solo 4 % de falsos positivos (289 tweets).

Los resultados obtenidos muestran que es posible identificar Eventos de la ciudad asociados al dominio para la mayoría de los tweets recuperados completos y correctamente escritos. Para estos Eventos es posible contar con una ubicación geográfica precisa asociada a elementos del mobiliario urbano que son el objeto de la queja o reclamo.

Un análisis individual de los tweets identificados incorrectamente como Eventos (falsos positivos) muestra que el 50 % están etiquetados como POCO_UTIL y UTIL, es decir tratan sobre limpieza en general pero no como reclamo concreto. También se pueden encontrar en este grupo gran cantidad de tweets sobre limpieza como noticia, o interacciones con este tipo de tweets. Si bien según los criterios definidos para este trabajo es un error identificar estos tweets como eventos, estos tienen interés para otras aplicaciones, como puede ser el análisis de posicionamiento del servicio entre los usuarios de la red social y en los medios de comunicación.

Los problemas de georreferenciación, identificación temática e identificación de imágenes fueron enfrentados exitosamente mediante módulos independientes basados en el estado del arte de diversas tecnologías. Esperamos que la decisión de liberar el código emanado de este trabajo y su naturaleza modular permitan su extensión y reutilización por parte de otros estudiantes e investigadores de manera de resolver los problemas que persisten y obtener mejores resultados.

La identificación de eventos, ubicados en el tiempo y el espacio, a partir de la perspectiva diversa de los ciudadanos y con información agregada como fotografías o videos permiten un nivel de análisis que posibilitaría, por ejemplo, la detección de zonas problemáticas, la identificación de patrones, o el análisis de sentimiento para determinado servicio público.

Si la plataforma fuera parte de una política institucional junto a campañas de comunicación, la recuperación de tweets podría ser mejor dirigida y junto a otras fuentes de información podría lograr alcanzar a una mayor cantidad de ciudadanos.

Dando un paso más, si cada Evento identificado por el sistema pudiera ser asignado directamente a un equipo para su resolución, y ese equipo pudiera reportar el resultado mediante imágenes directamente a la plataforma, se lograría cerrar el círculo de comunicación.

Es en este contexto de avances tecnológicos, y luego de analizar los resultados arrojados por este trabajo que podemos concluir que es posible construir mecanismos de sensado de redes sociales que puedan convertirse en un nuevo tipo de interfaces entre ciudadanos y organizaciones públicas. Interfaces que permitan una mayor participación de los ciudadanos en un terreno que les es familiar: las redes sociales, y que ofrezcan una resolución más

transparente de algunos reclamos. Nuevas interfaces que colaboren en la transformación de una ciudad en una ciudad inteligente.

Finalmente, otro aspecto a destacar es que en el estudio del estado del arte de trabajos relacionados no se encontraron aplicaciones similares a los efectos de comparar resultados.

7.2. Trabajo a futuro

Las propuestas para trabajo futuro son de dos tipos. Por un lado posibles mejoras a los diferentes módulos que componen el sistema. Por otra parte, funcionalidades que podrían ser agregadas al sistema para su funcionamiento en un contexto no académico.

7.2.1. Módulos

Aquí se presentan las diferentes mejoras que se les podría realizar a los módulos que componen al sistema para obtener mejores resultados que los mostrados en el Capítulo 6.

7.2.1.1. Módulo de procesamiento de imágenes

Como se explicó en 5.3.5 al momento de implementar este módulo se tenían dos posibilidades, optándose por la que utilizaba la Cloud Vision API provista por Google. Se decidió por esta, por el hecho de que no se contaba con el tiempo ni con los recursos necesarios como para construir un *corpus* de imágenes de Montevideo que permitiera entrenar un modelo propio. Se plantea como posible mejora al módulo entrenar un modelo a partir de un *corpus* personalizado según dominio y ciudad que permita obtener resultados más ajustados. En el caso de este proyecto, el entrenamiento de un modelo propio hubiera permitido detectar imágenes que tuvieran contenedores de Montevideo, por más que no hubiera presencia clara de basura.

Por otro lado, observando los falsos negativos obtenidos para el módulo desarrollado, se puede ver que las imágenes contienen basura pero que se encuentra en un espacio muy pequeño de la imagen. La mejora que se plantea es, para estos casos, realizar algunas transformaciones en la imagen que permitan destacar la basura en ella.

7.2.1.2. Módulo de información gramatical

La mejora planteada es lograr anotar un *corpus* lo suficientemente grande como para poder entrenar un modelo para el reconocimiento de entidades con nombre (NER) específico para tweets en español. La anotación del *corpus* es compleja pero podría realizarse en un esfuerzo colaborativo utilizando la herramienta que se describe en el Apéndice A.

7.2.1.3. Módulo de georreferenciación

Como se mencionó en 5.3.4 actualmente se utilizan todos los *tokens* del texto de la Pieza de Información para buscar en las diferentes colecciones de ubicaciones. En el marco de este trabajo, esto no es un problema para la performance por el hecho de que la longitud de un tweet está acotada a 140 caracteres. En caso de que la fuente de información sea una diferente a Twitter, como pueden ser blogs o correos electrónicos que no tienen cota de longitud, el enfoque tomado para la implementación del Módulo de georreferenciación puede pasar a ser un problema para la performance del sistema.

Se plantea como una posible mejora, disminuir la cantidad de *tokens* para los cuales se busca en las colecciones de ubicaciones. Para disminuir esta cantidad, el cambio puede estar en incluir al flujo que recorre una Pieza de Información el Módulo de información gramatical y utilizar los *tokens* etiquetados como ubicación para buscar en las colecciones utilizadas y no el resto. Alternativamente, podrían considerarse solo aquellos *tokens* que hayan sido etiquetados como sustantivos por el método de etiquetado POS que incluye el Módulo de información gramatical.

Estas mejoras ayudarían a disminuir la cantidad de *tokens* a buscar, por lo que mejoraría la performance del sistema. Adicionalmente, al disminuir la cantidad de palabras a buscar, disminuiría la cantidad de posibles errores a la hora de encontrar la ubicación exacta referida en el texto mejorando los resultados tanto del módulo como del sistema en general.

Otra posible mejora es utilizar N-gramas de longitud variable para buscar en las colecciones de ubicaciones utilizadas. Esto impactaría en el caso de que una ubicación esté referida por más de una palabra (por ej. Pablo María) lo que disminuiría la cantidad de falsos positivos encontrados por el módulo.

Para el caso de ubicaciones con numeración, sería posible integrar a la búsqueda de intersecciones estrategias que ubiquen la numeración en el caso en el que se encuentra a continuación del nombre de una calle.

Por último para el caso de los errores ortográficos, se plantea como mejora definir una distancia mínima de edición y construir todas las palabras que cumplan con esta distancia para cada una de las palabras que forman las colecciones de ubicaciones, agregándolas a los índices de texto provistos por MongoDB. Esto permitiría, ante una cierta cantidad (menor o igual a la distancia elegida) de errores ortográficos, poder detectar correctamente la ubicación referida en el texto de la Pieza de Información.

7.2.1.4. Módulos clasificadores de dominio

Ambos módulos clasificadores de dominio, tanto el Módulo clasificador de dominio para reclamos como el Módulo clasificador de dominio para limpieza se construyeron a partir de un *corpus* que contenía reclamos provistos por la IM. Estos reclamos, como se mencionó en la Sección 5.3, dada su longitud, se pueden tomar como una aproximación a tweets. Pudiendo construir un *corpus* de una magnitud similar al de reclamos pero compuesto exclusivamente de tweets, los resultados obtenidos por ambos clasificadores llegarían a ser mejores ya que, por ejemplo, el léxico utilizado en Twitter no siempre es el mismo al utilizado al realizar un reclamo formal a una institución como la IM.

7.2.1.5. Nuevos módulos

Como trabajo futuro en el marco de este proyecto se propone, como forma de disminuir la cantidad de falsos positivos al momento de detectar un Evento, construir un nuevo módulo clasificador cuyo objetivo sea decidir si una Pieza de Información corresponde a una noticia.

Por otra parte sería interesante conocer el posicionamiento del servicio entre los usuarios de la red social y los medios de comunicación así como conocer si el pensamiento de la población sobre el servicio es positivo o negativo. La implementación de módulos para estas tareas podrían relacionarse a otros proyectos de grado enfocados al análisis de sentimientos.

7.2.2. Funcionalidades

En un contexto no académico se le podrían agregar al sistema diferentes funcionalidades, las cuales son explicadas en esta sección.

7.2.2.1. Eventos incompletos

Muchas veces los tweets son reclamos de limpieza pero no se les puede reconocer la ubicación, por ejemplo a causa de errores en el nombre de una calle o directamente su ausencia. La funcionalidad que se plantea como trabajo futuro es lograr identificar estos tweets y responderlos automáticamente pidiendo una corrección o la información faltante, de forma tal de lograr identificar tanto el dominio del reclamo como también su ubicación.

7.2.2.2. Asignación de eventos

Sería interesante poder completar el ciclo del evento, desde su generación hasta su resolución. Para ello, se plantea la posibilidad de poder asignar un Evento o un conjunto de Eventos a los encargados de resolverlos. Una vez resuelto, registrar la resolución del problema y notificar vía Twitter a los denunciantes del evento.

7.2.2.3. Campañas de denuncia

El estado del arte en reconocimiento de imágenes permite la identificación de texto en ellas con facilidad. Para el caso de la limpieza se plantea como posible extensión, el hecho de agregar a los contenedores una imagen con un código que los identifique junto con un mensaje, promoviendo una nueva vía para realizar denuncias, ya sea cierto *hashtag* o mencionando alguna cuenta de Twitter dedicada exclusivamente para ello. El sistema podría detectar un Evento a partir del *hashtag* (o la cuenta dedicada) y reconocer el contenedor específico a partir del código en la imagen. De esta forma podrían centralizarse los reclamos de limpieza y así facilitar tanto el trabajo de procesarlos como también de los ciudadanos a la hora de realizarlos.

7.2.2.4. Conexión con otros sistemas

Si bien este trabajo se centraliza únicamente en el dominio de limpieza, se podría extender a más áreas. El sistema tomaría los diferentes reclamos realizados por los ciudadanos a través de Twitter y trataría de agruparlos según el dominio al que correspondan. A su vez, podría conectarse con el sistema SUR y así permitir que los ciudadanos hagan el seguimiento de sus reclamos. Se lograría sumar una nueva interfaz a SUR para los diferentes reclamos que llegan a través de Twitter logrando que el ciudadano realice un reclamo de manera más ágil.

7.2.2.5. Eliminación de falsos positivos

Sería interesante, a medida que se visualizan los Eventos detectados por la aplicación, eliminar aquellos que no representan un Evento de forma tal de ir disminuyendo la cantidad de falsos positivos que se visualizan en el sistema. Si bien esta funcionalidad no supone ninguna dificultad para realizarse, en el marco de este proyecto se decidió por no realizarla debido a que, a la hora de evaluar lo implementado, resultaba interesante tener la cantidad exacta de falsos positivos generados por el sistema.

Glosario

Árbol de Decisión Son un método no paramétrico de aprendizaje supervisado utilizado tanto para clasificación como para regresión (2.2.4.1). 15, 24, 59, 60

Algoritmos Greedy Algoritmos que siguen heurísticas basadas en óptimos locales. 22

API Interfaz de programación de aplicaciones, abreviada API del inglés: Application Programming Interface, es un conjunto de métodos de comunicación bien definidos entre componentes de software difrentes. 6, 7, 23, 36–38, 42, 43, 45, 46, 62

Bag of Words Técnica en la que un documento es representado como un conjunto de términos. 21, 34, 85, 86

Ciudad Inteligente Concepto relacionado al uso de la información y la tecnología disponible en una ciudad para la resolución de problemas derivados del crecimiento demográfico, de infraestructura, etc.. 1, 5, 25, 29, 81

Clasificador Naive Bayes Es un clasificador probabilístico fundamentado en el teorema de Bayes. 24

Corpus Conjunto amplio y estructurado de datos que generalmente representan ejemplos de uso de una lengua en un contexto. 7, 12–15, 21–24, 27, 30, 33, 34, 36, 43, 44, 46, 47, 51, 54–59, 69, 75, 76, 81, 82

CountVectorizer Transformador de Scikit-Learn que represenata un documento como *Bag of words*. 57, 59

Cross Validation Proceso de evaluación para clasificadores en el que se divide el corpus de entrenamiento en K partes iguales. Se entrena sobre $(k-1)$ partes y se evalúa sobre la parte restante. Se repite el proceso para todas las partes y se calcula la media. 15, 58, 71, 73, 75

CSV Archivo de valores separadas por coma. 56

Entropía La entropía en el contexto de los árboles de decisión caracteriza la impureza de un conjunto de instancias. 75

Evento Actividad del mundo real que ocurre durante cierto período de tiempo en cierto espacio geográfico. 3, 5, 6, 12, 21–25, 27–29, 37–40, 44, 47, 59–62, 64, 68, 69, 75–77, 80, 82, 83

Fall-out Se puede ver como la probabilidad de clasificar positivamente instancias negativas. Se calcula como el cociente entre la cantidad de instancias que se clasificaron incorrectamente y la cantidad de instancias que no pertenecen a la clase. 14, 76

Features Características o rasgos de un documento que lo definen. 22–24, 59, 60, 75, 88

Framework Es una abstracción en la que el software que provee un conjunto de funcionalidades genéricas puede modificarse fácilmente para proveer funcionalidades especificas. 24, 33, 36

Gain La reducción de la entropía causada por el particionado de instancias según el atributo que se está evaluando. 75

Geohash Codificación de coordenadas geográficas cuya representación es una palabra y representa un rectángulo en la superficie de la tierra. 48

GPS Sistema de posicionamiento Global. 25

GridSearchCV Es una herramienta de Scikit-Learn que permite realizar el ajuste sobre distintos parámetros eligiendo la mejor combinación al finalizar el proceso. 57–60

Hashtag Etiqueta de metadatos usada en redes sociales como Twitter. 24, 25, 48, 83

HCA Técnicas de *clustering* jerárquico. 22

IA Inteligencia Artificial. 9, 37, 79

IE Information Extraction, Extracción de Información. 12

IM Intendencia de Montevideo. 25, 27, 28, 56, 59, 87

IR Information Retrieval, Recuperación de Información. 11

K-means Algoritmo de clustering que agrupa los diferentes documentos en K grupos utilizando la media como función de distancia. 22

K-median Algoritmo de clustering que agrupa los diferentes documentos en K grupos utilizando la mediana como función de distancia. 22, 86

K-median++ Variante del algoritmo K-median. 22

LinearSVC Implementación del algoritmo SVM para la librería Scikit-Learn. 57

Mixed Vectors Representación de documentos que integra los conceptos de *term vectors*,*bag of words* y *named entity vectors*. 22

Named Entity Vectors Representación de documentos que intenta extraer información contestando las preguntas Quién, Cómo, Cuándo y Dónde. 22, 86

NED Detección de nuevos Eventos. 22, 28

NER Reconocedor de entidades con nombre de Stanford. 24

NER Reconocimiento de Entidades con Nombre. 12, 34, 46, 47, 81

Participatory Sensing Recolección de información local, participativa y en tiempo real a partir de los dispositivos móviles conectados a Internet de los ciudadanos. 1, 25

Pieza de Información Documento candidato a convertirse en evento, que contiene los datos originales de la fuente de información además de varios campos de control. 34, 35, 38–40, 43, 45–47, 51, 54, 59, 60, 68, 69, 75, 81, 82

PLN Procesamiento del Lenguaje Natural. 6, 7, 9–11, 13, 16, 18, 36

POS Etiquetado gramatical de palabras. 17, 34, 46, 47, 82

Precision Se puede considerar como una medida de la exactitud de los clasificadores. Se calcula como el cociente entre la cantidad de instancias que se clasificaron correctamente y la cantidad de instancias que se clasificaron como pertenecientes a la clase. 13, 14, 58, 59, 71, 74, 77

Recall Se puede considerar como una medida de la completitud de los clasificadores. Se calcula como el cociente entre la cantidad de instancias que se clasificaron correctamente y la cantidad de instancias que pertenecen a la clase. 13, 14, 25, 58–60, 71, 73–77

RED Detección retrospectiva de Eventos. 22, 28

Retweet Forma de compartir un tweet ajeno por parte de usuarios de Twitter. 24, 43

RRSS redes sociales. 5, 6, 12, 19, 21–23, 26, 27, 29, 79, 80

RSS Really Simple Syndication. 23

Snapshots Se definen como ternas compuestas de una entidad objetivo, un período de tiempo, y un conjunto de tweets sobre la entidad en ese período. 24

Stemming En español lematización, el proceso de reducir las formas inflexionadas y las formas derivadas de una palabra a una forma básica común. 48

Stopword Son palabras que no agragan ninguna información al texto desde un punto de vista semántico. 18, 35, 47

SUR Sistema Único de Reclamos de la IM. 25, 29, 56, 58, 71, 74, 83

SVC Implementación del algoritmo SVM para la librería Scikit-Learn que permite la predicción de probabilidades. 59

SVM Support Vector Machines. 16, 17, 24, 57–59, 86, 87

Term Vectors Técnica en la que un documento es representado como un vector de términos. 21, 24, 34, 86

Tf-Idf Es una medida numérica que expresa cuán relevante es un término para un documento dentro del corpus. 22, 24, 25, 57, 88

TfIdfTransformer Transformador de Scikit-Learn que representa un documento utilizando Tf-Idf. 57, 59

Token Los tokens representan palabras o números. 17, 46–48, 50, 81

Tweet Mensaje de hasta 140 caracteres que los usuarios de Twitter comparten con sus seguidores. 7, 23–25, 28, 30, 34, 36–38, 40–47, 51, 54–56, 58, 59, 61–63, 65, 67, 69, 71, 80, 81, 83

Técnicas Document-Pivot Familia de técnicas que agrupan documentos en base a su similitud textual. 21

Técnicas Feature Pivot Técnicas en las que los documentos son representados como un conjunto de palabras clave que lo caracterizan o *Features*. 22

Apéndice A

Herramientas para Stanford

Se desarrollaron algunas herramientas para trabajar con los etiquetadores NER y POS de Stanford, que se encuentran bajo una licencia libre en `https://github.com/raulsperoni/stanford-taggers-tools` disponibles para su uso y modificación. Estas herramientas finalmente no fueron utilizadas para este trabajo, y por lo tanto no fueron probadas adecuadamente. Sin embargo pueden ser de utilidad para quienes quieran interactuar con los parsers de Stanford de manera más sencilla.

A.1. Servidor de descarga de tweets

Permite la descarga de tweets utilizando una de las API de Twitter y su almacenamiento en una base de datos MongoDB. Los criterios de búsqueda se setean vía variables de Entorno y las credenciales de Twitter se cargan desde el archivo libs/config.py.

A.2. Servidor de entrenamiento

Permite mediante servicios rest la anotación de un corpus de tweets con etiquetas NER. El servicio está autenticado y permite crear nuevos usuarios con un pin que se setea vía variable de entorno. El desarrollo es con Python y Flask.

A.3. Web de entrenamiento

Permite mediante una web desarrollada en AngularJS anotar cada uno de los tweets descargados con etiquetas NER como se ve en la Figura A.1.

Las etiquetas pueden ser:

- LUG: La etiqueta para marcar nombres de ubicaciones geográficas, ya sean calles o lugares identificables. Si la ubicación consta de varias palabras se deben marcar todas. Ej. Ciudad Vieja, Luis Alberto de Herrera
- PER: Etiqueta para marcar personas físicas, puede ser un nombre propio o un nombre de Twitter con @. Ej. Daniel Martínez, @dMartinezUy
- ORG: Etiqueta para marcar organizaciones, puede ser un nombre propio o un nombre de Twitter con @. Ej. Intendencia de Montevideo, @montevideoim
- OTRA: Etiqueta para marcar una entidad no perteneciente a las clases anteriores.

Figura A.1: Ejemplo de anotación de tweets

A.4. Jupyter

Levanta un servidor Python Jupyter para interactuar con los demás módulos y scripts. Es posible utilizando notebooks generar un archivo tabulado a partir del corpus anotado vía web. Este archivo será la entrada para entrenar nuevos modelos NER para el etiquetador de Stanford.

A.5. Servidor NER y POS

Expone mediante servicios rest métodos para:

- Elegir de entre los modelos e idiomas disponibles.
- Cargar un nuevo corpus anotado, y entrenar un nuevo modelo.
- Recuperar etiquetas POS para un texto dado.
- Recuperar etiquetas NER para un texto dado.

El desarrollo es con Python y Flask pero interactúa con procesos Java que son los etiquetadores de Stanford en ejecución.

Bibliografía

[1] Accan. Introduction to social media. `http://accan.org.au/files/Tip%20Sheets/Introduction%20to%20Social%20Media.pdf`. [Online; Último acceso 26-Enero-2017].

[2] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. 1998.

[3] Samar M Alqhtani, Suhuai Luo, and Brian Regan. Fusing text and image for event detection in twitter. *arXiv preprint arXiv:1503.03920*, 2015.

[4] Farzindar Atefeh and Wael Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164, 2015.

[5] Hila Becker, Mor Naaman, and Luis Gravano. Selecting quality twitter content for events. *ICWSM*, 11, 2011.

[6] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[7] Christos Bouras and Vassilis Tsogkas. Assigning web news to clusters. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, pages 1–6. IEEE, 2010.

[8] Mauricio Bouskela. La ruta hacia las smart cities. `http://goo.gl/3Y0tWK`, Enero 2017. [Online; Último acceso 17-Enero-2017].

[9] Jeffrey A Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya Ramanathan, Sasank Reddy, and Mani B Srivastava. Participatory sensing. *Center for Embedded Network Sensing*, 2006.

[10] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM, 2011.

[11] Hafedh Chourabi, Taewoo Nam, Shawn Walker, J Ramon Gil-Garcia, Sehl Mellouli, Karine Nahon, Theresa A Pardo, and Hans Jochen Scholl. Understanding smart cities: An integrative framework. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 2289–2297. IEEE, 2012.

[12] Hinrich Schütze Christopher D. Manning, Prabhakar Raghavan. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[13] Intendencia de Montevideo. Datos abiertos. http://www.montevideo.gub.uy/institucional/montevideo-abierto/datos-abiertos, Abril 2015. [Online; Último acceso 14-Abril-2016].

[14] Intendencia de Montevideo. Montevideo presenta experiencia de gobierno electrónico. http://www.montevideo.gub.uy/institucional/noticias/montevideo-presenta-experiencia-de-gobierno-electronico, Marzo 2015. [Online; Último acceso 24-Marzo-2016].

[15] Intendencia de Montevideo. Ciudades inteligentes para la inclusión y la sostenibilidad. http://www.montevideo.gub.uy/servicios-y-sociedad/ciudades-inteligentes-para-la-inclusion-y-la-sostenibilidad, Enero 2016. [Online; Último acceso 1-Enero-2017].

[16] Grupo de PLN InCo Facultad de Ingeniería Udelar Uruguay. Análisis léxico. https://eva.fing.edu.uy/course/view.php?id=211, 2016. [Online; Último acceso 1-Enero-2017].

[17] Grupo de PLN InCo Facultad de Ingeniería Udelar Uruguay. Clasificación. https://eva.fing.edu.uy/course/view.php?id=211, 2016. [Online; Último acceso 1-Enero-2017].

[18] Grupo de PLN InCo Facultad de Ingeniería Udelar Uruguay. Introducción al procesamiento de lenguaje natural. https://eva.fing.edu.uy/course/view.php?id=211, 2016. [Online; Último acceso 1-Enero-2016].

[19] Grupo de PLN InCo Facultad de Ingeniería Udelar Uruguay. Morfología. https://eva.fing.edu.uy/course/view.php?id=211, 2016. [Online; Último acceso 1-Enero-2017].

[20] Grupo de PLN InCo Facultad de Ingeniería Udelar Uruguay. Recuperación de información. https://eva.fing.edu.uy/course/view.php?id=211, 2016. [Online; Último acceso 1-Enero-2017].

[21] ICT Facts. Figures-the world in 2015. *Geneva: The International Telecommunication Union (ITU)*, 2015.

[22] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[23] Anthony Fox, Chris Eichelberger, James Hughes, and Skylar Lyon. Spatio-temporal indexing in non-relational distributed databases. In *Big Data, 2013 IEEE International Conference on*, pages 291–299. IEEE, 2013.

[24] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[25] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S Yu, and Hongjun Lu. Parameter free bursty events detection in text streams. In *Proceedings of the 31st international conference on Very large data bases*, pages 181–192. VLDB Endowment, 2005.

[26] Miguel Ángel García Cumbreras, Eugenio Martínez Cámara, Julio Villena Román, and Janine García Morera. Tass 2015–the evolution of the spanish opinion mining systems. 2016.

[27] gate.ac.uk. Introduction to named entity recognition - gate. `https://gate.ac.uk/sale/talks/stupidpoint/diana-fb.ppt`. [Online; Último acceso 1-Enero-2017].

[28] Saurabh Goorha and Lyle Ungar. Discovery of significant emerging trends. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–64. ACM, 2010.

[29] Qi He, Kuiyu Chang, and Ee-Peng Lim. Analyzing feature trajectories for event detection. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 207–214. ACM, 2007.

[30] Qi He, Kuiyu Chang, Ee-Peng Lim, and Jun Zhang. Bursty feature representation for clustering text streams. In *SDM*, pages 491–496. SIAM, 2007.

[31] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[32] Jonathan Hurlock and Max L Wilson. Searching twitter: Separating the tweet from the chaff. In *ICWSM*, pages 161–168, 2011.

[33] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[34] Roman Yangarber Jakub Piskorski. *Information Extraction: Past, Present and Future*. Springer, 2012.

[35] Jon Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[36] April Kontostathis, Leon M Galitsky, William M Pottenger, Soma Roy, and Daniel J Phelps. A survey of emerging trend detection in textual data mining. In *Survey of text mining*, pages 185–224. Springer, 2004.

[37] Giridhar Kumaran and James Allan. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304. ACM, 2004.

[38] Kyumin Lee, Brian David Eoff, and James Caverlee. Seven months with the devils: A long-term study of content polluters on twitter. In *ICWSM*, 2011.

[39] Kalev Leetaru, Shaowen Wang, Guofeng Cao, Anand Padmanabhan, and Eric Shook. Mapping the global twitter heartbeat: The geography of twitter. *First Monday*, 18(5), 2013.

[40] Gang Luo, Chunqiang Tang, and Philip S Yu. Resource-adaptive real-time new event detection. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 497–508. ACM, 2007.

[41] Michael Mathioudakis and Nick Koudas. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM, 2010.

[42] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science, 1997.

[43] MongoDB. The mongodb 3.2 manual. `https://docs.mongodb.com/manual/`, Noviembre 2016. [Online; Último acceso 5-Noviembre-2016].

[44] Eduardo Moreno. *Urbanization and development : emerging futures*. UN Habitat, New York, 2016.

[45] Mor Naaman, Hila Becker, and Luis Gravano. Hip and trendy: Characterizing emerging trends on twitter. *Journal of the American Society for Information Science and Technology*, 62(5):902–918, 2011.

[46] Masayuki Okamoto and Masaaki Kikuchi. Discovering volatile events in your neighborhood: local-area topic extraction from blog entries. In *Asia Information Retrieval Symposium*, pages 181–192. Springer, 2009.

[47] Ron Papka. On-line new event detection, clustering, and tracking. Technical report, DTIC Document, 1999.

[48] Swit Phuvipadawat and Tsuyoshi Murata. Breaking news detection and tracking in twitter. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 3, pages 120–123. IEEE, 2010.

[49] Ana-Maria Popescu and Marco Pennacchiotti. Detecting controversial events from twitter. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1873–1876. ACM, 2010.

[50] Ana-Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. Extracting events and event descriptions from twitter. In *Proceedings of the 20th international conference companion on World wide web*, pages 105–106. ACM, 2011.

[51] Grupo Radar. Perfil internauta uruguayo 2016 - resumen ejecutivo. `http://www.gruporadar.com.uy/01/wp-content/uploads/2016/11/El-Perfil-del-Internauta-Uruguayo-2016-Resumen-Ejecutivo.pdf`, Enero 2017. [Online; Último acceso 17-Enero-2017].

[52] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[53] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[54] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 42–51. ACM, 2009.

[55] Scikit-Learn. Decision trees. http://scikit-learn.org/stable/modules/tree.html. [Online; Último acceso 26-Enero-2017].

[56] Scikit-Learn. Support vector machines. http://scikit-learn.org/stable/modules/svm.html. [Online; Último acceso 26-Enero-2017].

[57] Tristan Snowsill, Florent Nicart, Marco Stefani, Tijl De Bie, and Nello Cristianini. Finding surprising patterns in textual data streams. In *2010 2nd International Workshop on Cognitive Information Processing*, pages 405–410. IEEE, 2010.

[58] Stanford. The stanford natural language processing group. http://nlp.stanford.edu/software/, Octubre 2016. [Online; Último acceso 30-Octubre-2016].

[59] svms.org. Introduction - support vector machines. http://www.svms.org/introduction.html. [Online; Último acceso 26-Enero-2017].

[60] Mariona Taulé, Aina Peris, and Horacio Rodríguez. Iarg-ancora: Spanish corpus annotated with implicit arguments. *Lang. Resour. Eval.*, 50(3):549–584, September 2016.

[61] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[62] Stanford University. Stanford log-linear part-of-speech tagger. http://nlp.stanford.edu/software/tagger.shtml. [Online; Último acceso 26-Enero-2017].

[63] Stanford University. Stemming and lemmatization. http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html. [Online; Último acceso 26-Enero-2017].

[64] Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. Mining correlated bursty topic patterns from coordinated text streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 784–793. ACM, 2007.

[65] The Speech Recognition Wiki. The speech recognition wiki. http://recognize-speech.com/language-model/from-the-phonome-sequence-to-a-word/13-language-model, 2016. [Online; Último acceso 8-Febrero-2017].

[66] Wikipedia. Precision and recall. https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg. [Úlitmo acceso: 28-Enero-2017].

[67] Yiming Yang, Jaime G Carbonell, Ralf D Brown, Thomas Pierce, Brian T Archibald, and Xin Liu. Learning approaches for detecting and tracking news events. 2000.

[68] Yiming Yang, Tom Pierce, and Jaime Carbonell. A study of retrospective and online event detection. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 28–36. ACM, 1998.