UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

EXTRACCIÓN DE CONTEXTO GEOGRAFICO A PARTIR DE NLP PARA INFORMACIÓN DE TRÁNSITO EN REDES SOCIALES

Duvan Alexander Robles Mondragón

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Maestría en Ciencias de la Información y las Comunicaciones
Bogotá D.C., Colombia
2020

EXTRACCIÓN DE CONTEXTO GEOGRAFICO A PARTIR DE NLP PARA INFORMACIÓN DE TRÁNSITO EN REDES SOCIALES

Duvan Alexander Robles Mondragón

Trabajo de grado presentado como requisito parcial para optar por el título de:
Magister en Ciencias de la Información y las Comunicaciones

Director:
Ing. Cesar Andrey Perdomo MSc.

Grupo de investigación:
LASER

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Maestría en Ciencias de la Información y las Comunicaciones
Bogotá D.C., Colombia
2020

Memento Humani

Agradecimientos

Agradezco a quienes acompañaron este proceso de formación, creo fuertemente en el papel de la educación y de quienes se dedican a esto, por eso, gratitud con los compañeros y buenos profesores que me encontré en el camino, también a mi tutor que estuvo presente y apoyo de forma constante aún cuando los ánimos no eran los mejores, Gracias.

Resumen

Poco se ha hablado de la recuperación de información espacial de texto, en particular porque el termino "información espacial" se asocia con geometrías en forma de vectores o información de tipo raster que expresan distintas variables o fenómenos acompañados de coordenadas, pese a esto la extracción de información en texto se presenta como uno de los avances más prometedores gracias al procesamiento natural del lenguaje (NLP) y en este caso se perfila como un nuevo campo de acción complementario al análisis espacial intentando extraer un evento especifico que sucede en el espacio y se plasmó en un texto. La fuente principal de texto, para esta investigación, son los compartidos en una red de colaboración como twitter, Los eventos extraídos son los que se encuentran o hacen referencia a la malla vial y que afectan la movilidad de forma recurrente o aleatoria, este último, el aleatorio, el más difícil de manejar en una ciudad cualquiera que debe monitorear el tránsito de actores viales bajo una red de sensores que intentan ver la congestión de las vías e incidencias viales. Ahora bien, estos textos fueron almacenados bajo un esquema de base de datos clasificados como incidencia vial que se pasan sobre un reconocedor de patrones de escritura que extrae la localización y posteriormente alimenta un georreferenciador que devuelve un par de coordenadas (lat, lon), la idea con estas coordenadas es convertirlas en datos compilados que dentro de un análisis espacial muestren un fenómeno de agrupamiento bajo técnicas geoestadisticas como la autocorrelación espacial, encontrando puntos calientes o puntos fríos de existencia de incidentes.

Los resultados geográficamente definidos se comparan con datos de años recientes levantados por entidades oficiales de tránsito y que son publicados para el acceso libre, la comparación de patrones entre un año anterior y los extraídos con inteligencia artificial muestran comportamientos espaciales similares y la auto correlación espacial conserva cierta similitud dejando ver la utilidad de la extracción de foco geográfico que se plantea y posible complemento a fuente de datos para el manejo de la congestión vial e incidencias de tránsito.

Palabras clave

Inteligencia artificial, Recuperación de información, datos espaciales, procesamiento natural del lenguaje; máquinas de soporte vectorial, Geo codificación, Gazetteers, análisis espacial.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO DE TÉRMINOS

gazetteers Son llamados también diccionarios geográficos, contiene nombres de lugares, describe ríos, montañas, municipios, ciudades u otros elementos geográficos.

kernel abstracción de filtro de varias dimensiones usado en el uso de redes neuronales convolucionales.

PostGIS Extensión para manejo de bases de datos espaciales en PostgreSQL.

ACRÓNIMOS

AI Inteligencia artificial.

API Application programming interface.

CNN Redes neuronales convolucionales.

GIR Recuperación de información geográfica.

GPS Sistema de posicionamiento global.

GPT-3 Generative Pre-trained Transformer 3.

KNN k-nearest neighbors.

LAT Latitud.

LON Longitud.

LSTM Long Short-Term Memory.

MLP Perceptron multicapa.

NER Extracción de entidades nombradas.

NLP Procesamiento natural del lenguaje.

NLU Entendimiento del lenguaje natural.

PYSAL Python spatial analysis library.

ReLU Función lineal rectificada.

RNN Redes neuronales recurrentes.

SIG Sistemas de información geográfica.

SVM Máquinas de soporte vectorial.

TSP Problema del vendedor ambulante.

UPZ Unidad de planeamiento zonal.

Capítulo 1

INTRODUCCIÓN

Es claro que un dato por si mismo no da mucha información, pero cuando se convierte en un dato agregado empieza a tener sentido para el análisis o recuperación de información, también llamada minería de datos. Lo interesante es como la minería de datos empieza a ahorrar esfuerzos y recursos en las instituciones, si bien las aplicaciones son muchas el monitoreo de eventos configurados en el espacio requieren de una alta inversión en las etapas de instalación de sensores, recolección de datos, procesamiento y posterior análisis, a esto se suma el mantenimiento de la red de monitoreo y su vida útil teniendo que considerar adicionalmente una cobertura, pues los sensores no se encuentran posicionados muy cerca dentro de la red y cuya solución incluye el uso de algoritmos de ingeniería de datos para completar información donde no existen sensores. El presente documento plantea la recuperación de información de contexto geográfico como alternativa al monitoreo de incidencias viales usando como recurso principal las comunidades web de tráfico orgánico, dicha recuperación usa minería de texto y procesamiento natural del lenguaje (NLP) para la preparación de datos y posterior entrenamiento que permite clasificar incidencias sobre vías en la ciudad de Bogotá.

El entendimiento del lenguaje natural toma como tarea principal la extracción de contexto geográfico sobre los datos clasificados evitando la ambigüedad en localización, por lo que un algoritmo especifico hará uso de extracción de información de entidades, “Named Entity Extraction” (NER) y recuperación de información geográfica (GIR) que alimentan un georeferenciador dedicado para Bogotá y del que se espera que los datos localizados sigan los mismos patrones geográficos que se levantan sobre la ciudad en años anteriores, lo que puede indicar que esta extracción de datos de bajo costo puede ser un complemento al monitoreo de ciudades como la capital Colombiana.

1.1. Planteamiento del problema

El procesamiento natural del lenguaje (NLP) es un área de investigación en ciencias computacionales e inteligencia artificial (AI) que se enfoca en procesar lenguaje natural como el inglés, español o Mandarín, entre otros (Lane y col., 2019). Este proceso generalmente involucra una traducción del lenguaje a datos numéricos que el computador pueda usar para entender el mundo, entonces los sistemas de NLP son semejantes a tuberías pues dentro hay un flujo de varios estados de procesamiento en donde el lenguaje fluye para ser procesado y extraer así información.

Las expresiones regulares usan un tipo especial de gramática formal llamada gramática regular, la gramática regular tienen un comportamiento predecible y demostrable permitiendo la recuperación de información, haciendo posible estructurar información de textos no estructurados. Esta extracción de información hace parte de un área de investigación llamada entendimiento del lenguaje natural (NLU), en los casos en que se desee extraer piezas de información, pero con foco geográfico se relaciona a la recuperación de objetos, en inglés "Named Entity Extraction" (NER) y en paralelo "Geographic Information Retrieval" (GIR).La identificación del foco geográfico de un texto consiste en determinar la principal o principales localizaciones a las que hace referencia un texto. (Peregrino y col., 2013)

Obtener el contexto geográfico es fundamental a la hora de crear un sistema automatizado que muestre una relativa importancia espacial ante un evento, además posibilita la identificación de patrones espaciales sobre variadas temáticas, por ejemplo, sitios turísticos, nombres, fechas, precios o incidencias viales. Actualmente gran parte de la información generada es no estructurada y se encuentra en forma de texto en la web, en su mayoría se comparte en sistemas de colaboración abierta, por ejemplo, en redes sociales, blogs, artículos, entre otros. Los textos que se comparten tienen diferentes motivaciones y algunos informan a la comunidad sobre eventos que son susceptibles de ser georeferenciados, este es el caso de eventos sobre la malla vial urbana que generan congestión. La congestión en redes viales representa uno de los mayores desafíos pues se comporta de formas distintas dependiendo la localización espacial, por lo que se ha establecido entonces divisiones por comportamiento y causa, el tráfico no recurrente se enfoca en incidentes viales que provocan congestión y el tráfico recurrente hace referencia a patrones regulares que generalmente ocurren en un sitio especifico, camino o área fija durante un periodo de tiempo que usualmente corresponde a horas pico (An y col., 2016), es común su ocurrencia debido a la alta demanda de transporte, capacidad de tráfico insuficiente o pobre infraestructura. El tráfico recurrente afecta la ciudad en la escala regional por lo que se espera un patrón evolutivo, entonces identificar el patrón evolutivo permite diseñar estrategias

de mejoramiento del tráfico que beneficien la movilidad de una ciudad.

El monitoreo de la congestión vial es un reto para las entidades estatales pues la principal forma de hacer este control requiere el desarrollo de complejas estructuras de sensores posicionadas en algunos puntos estratégicos de las ciudades, esta instalación además de limitar la cobertura espacial de control es costosa, por lo tanto su precisión es limitada, sabiendo esto el aprovechamiento de información de texto en redes sociales se convierte en una alternativa de datos de bajo costo, crear sistemas de recuperación de información geográfica para ubicar incidencias viales e identificar patrones espaciales en los datos soluciona en parte la recolección de información y evidencia la necesidad de técnicas de procesamiento natural del lenguaje, NLP.

El problema consiste en establecer la extracción de contexto geográfico apoyado en técnicas NLP, entendimiento del lenguaje natural (NLU) y recuperación de información geográfica (GIR) con patrones regulares sobre información clasificada como incidencia vial determinando tasa de recuperación de información geográfica y distribución espacial.

1.2. Formulación del problema

Los interrogantes abordados son los siguientes:

1. ¿Se proporciona más información a los sistemas de detección de incidencias viales a partir de información espacial recuperada en fuentes no estructuradas con ayuda de procesamiento natural del lenguaje en lugar de usar redes convencionales de monitoreo?

2. ¿Cuáles son las ventajas y limitaciones del procesamiento del lenguaje natural para acceder a focos geográficos?

1.3. Objetivos

Este trabajo se enfoca en la extracción de contexto geográfico a partir de clasificaciones de incidencias viales en textos presentes en la ciudad de Bogotá y representar sus características con ciencia de datos espaciales.

1. Clasificar información no estructurada de sistemas de colaboración abierta a partir de un conjunto definido de características o grupo de entrenamiento.

2. Segmentar los datos capturados identificando incidencias viales susceptibles a ubicación espacial.

3. Aplicar técnicas de minería de datos que permitan identificar patrones en información espacial.

4. Realizar seguimiento a los patrones viales que permiten explicar correlaciones en la dinámica de movilidad Urbana de Bogotá.

Capítulo 2

MARCO DE REFERENCIA

2.1. Procesamiento Natural del lenguaje

El lenguaje natural dista mucho del lenguaje de programación, pues no está destinado a ser traducido a un número finito de operaciones matemáticas, esto se debe principalmente a que el lenguaje natural se usa para compartir información entre seres humanos, es decir un lenguaje de programación le dice al computador exactamente lo que debe hacer, pero no existen compiladores para idiomas como el inglés, español o francés. Entonces el procesamiento natural del lenguaje (NLP) es un área de investigación en ciencias de la computación e inteligencia artificial (IA) que se relaciona con el procesamiento de lenguajes naturales como el inglés o español, ese proceso implica traducir el lenguaje natural en números que una computadora puede entender, clasificar o usarse en la generación de nuevo texto (Karimzadeh y col., 2019). En el procesamiento natural del lenguaje las mismas tareas que hace diariamente el ser humano se llevan a cabo con ponderación, fragmentación, etiquetado de voz, traducción automática o reconocimiento de voz, todas estas tareas representan un gran reto computacional

La historia documentada del procesamiento natural del lenguaje inicia a finales de 1940, se han definido 4 fases importantes del NLP y sus aplicaciones, la primera fase corresponde efectivamente a la década de 1940 hasta 1960, la segunda desde finales de los sesenta (1960) hasta los setenta (1970), una tercera fase va hasta los ochenta y una cuarta hasta finales de los noventas (Jones, 1994). Los primeros años de investigación se enfocaron en máquinas traductoras esto es documentado en las investigaciones de Richen's y Weaver's, donde se mencionan las sutiles diferencias que existen entre el inglés americano y el inglés británico que hacen posible la comunicación entre nativos de los dos lenguajes, sin embargo acotan que existen mayores dificultades cuando se consideran idiomas de raíz menos relacionada (Joos y col.,

1956). Hacia 1954 IBM muestra un experimento de traducción automática desde el ruso al inglés de forma rudimentaria, este experimento hace que se activen en 1954 las instituciones académicas fundando así la revista "Mechanical translation" que años después se convertiría en "Computacional Linguistics" promoviendo conferencias alrededor de la traducción de textos.

Esta primera etapa fue un periodo para entusiastas e investigadores pues la traducción de textos hoy en día sigue siendo un campo de constante investigación en cuanto a la semántica y variedad lingüística, a esto se sumaba que para la década de los sesentas los recursos informáticos disponibles eran tarjetas perforadas, procesamiento por lotes, almacenamiento limitado y el acceso a máquinas era restringido, pese a esto hubo actividad en investigación internacional en zonas como la unión soviética, Estados Unidos, Europa y un poco de actividad en Japón. Esta época puede resumirse como la traducción por búsqueda pues el procesamiento se hacía palabra por palabra basada en diccionarios y se crearon nuevas estrategias para evitar la ambigüedad (Jones, 1994).

La segunda etapa tuvo un poco del conocimiento de la inteligencia artificial más generalizada en el mundo, el trabajo pionero que incluyo inteligencia artificial era un sistema de preguntas y respuestas llamado "BASEBALL", este era un programa que respondía preguntas sobre datos almacenados (Green y col., 1961), para esta etapa la investigación se centró en la semántica y en la comprensión de texto a partir de algoritmos. Para la tercera etapa del procesamiento natural del lenguaje, finales de los años setentas y finales del 80, esta se puede generalizar como la evolución del estilo gramático lógico impulsado en su momento por la teoría gramatical de los lingüistas durante los años 70 y el uso de la lógica para representación de conocimiento y razonamiento en inteligencia artificial. Esta época es una transición de la gramática computacional a aplicaciones prácticas pues los lingüistas desarrollaron la teoría gramática. En cuanto a investigación tuvieron notable desarrollo los sistemas de generación de texto de múltiples oraciones.

La última fase documentada por Karen Sparck Jones del laboratorio de computación de la universidad de Cambridge (Jones, 1994), indica que fue la época del lenguaje estadístico esto a finales de 1980 hasta finales del 2000, fue etiquetado como un periodo de recolección y codificación de datos. Hablando por primera vez de corpus de datos para identificación lingüística de patrones de concurrencia que podrían ser aplicados a la computación, sin duda la probabilidad y su enfoque se extendieron a través del procesamiento natural de lenguaje en parte por demostrar utilidad en procesamiento del habla y de ahí que se recomendara su uso e incluirlo como un modelo base de procesamiento NLP, para esta época aunque el NLP no era demasiado

útil si era usado en la recuperación de documentos a partir de textos, la inclusión obligó a que los sistemas fueran más allá de oraciones bien formadas y lidiaran con la ambigüedad en nombres propios. Para está época se avanzó fuertemente en el reconocimiento del habla con sistemas de dictado esto gracias a las técnicas de probabilidad y a el aumento del procesamiento de cómputo, como dice Jones, el NLP ha evolucionado durante 40 años y efectivamente ha tenido avances en unos pocos campos, sin embargo, se espera que las investigaciones contemporáneas retomen las primeras ideas a partir de los avances en léxico y en información estadística por lo menos en áreas de máquinas traductoras y recuperación de información.

Los últimos 20 después de lo documentado por Jones ha sido llamado procesamiento natural del lenguaje "moderno", dominado por métodos de transferencia de aprendizaje en particular basados en una arquitectura llamada "Transformers" que hacen parte del Deep learning y que mejoran algo llamado Long Short-Term Memory (LSTM), estos avances han sido vertiginosos desde 2019 pues han batido records de métricas, sin embargo los modelos y tecnologías son difíciles y costosos para investigadores o entusiastas, por ejemplo como se menciona en(Wolf y col., 2019) para la arquitectura RoBERTa, una red de clasificación de aprendizaje profundo que fue entrenada con 160 GB de texto sobre Amazon Web Services específicamente un servicio cloud, costó solo en entrenamiento aproximadamente 100.000 dólares, lo que sería inviable para un pequeño investigador. Del anterior problema la comunidad ha sido consiente de este problema y ha creado un nuevo framework para NLP llamado HuggingFace Transformes destinado a compartir modelos de alto desempeño con arquitecturas pre entrenadas.

2.2. Algoritmos de entendimiento del lenguaje

La era de la información exige el entendimiento no solo de datos estructurados, sino también de los no estructurados como el lenguaje, esto representa un reto para las industrias pues generamos terabytes de información en blogs, tweets, estados de Facebook, chats, correos electrónicos y comentarios en la web, esta información tiene un gran potencial pues puede mejorar la experiencia de los usuarios y alimentar nuevas ideas, actualmente existen muchos campos de aplicación de técnicas de procesamiento del lenguaje, algunas son; el deletreo, traducción, Motores de búsqueda, identificación de voz, clasificadores de correo, entre muchos otros (Perkins y col., 2016).

La clasificación es un problema importante en el procesamiento natural del lenguaje existiendo numerosas aplicaciones de clasificación como organizadores de do-

cumentos, filtrado de noticias, detección de spam, minería de opinión e identificación de sentimientos. Esto se puede lograr a través de métodos tradicionales como diccionarios de palabras buscadas en un texto que permiten categorizarlo, recientemente en estas tareas de clasificación se encuentran los modelos de aprendizaje profundo que incluyen las redes neuronales convolucionales (CNN) y redes neuronales recurrentes (RNN), estos modelos de aprendizaje profundo priorizan la localidad y secuencialidad, además de capturar la información semántica y sintáctica en secuencias de palabras consecutivas.

Sin embargo, antes de pensar en estas aplicaciones hay un orden lógico o un flujo de trabajo en NLP, recordemos que para enseñarle a un sistema de cómputo a entender palabras se debe encontrar una forma de traducir dichas palabras a un lenguaje que un computador pueda entender a esto se le llama vectorización (Tensorflow, 2019) y es fundamental a la hora de alimentar un proceso de NLP. En este caso hay varias formas de hacer dicha vectorización.

El primero se denomina "One-hot vector" cuya esencia de vectorización es tomar una frase por ejemplo "el perro está sentado en el sofá" y representar cada palabra única creando una matriz de ceros y unos con la longitud del número de palabras en el input este enfoque se puede ver en la siguiente tabla.

	el	perro	está	sentado	en	sofá
sofá	0	0	0	0	0	1
sentado	0	0	0	1	0	0
en	0	0	0	0	1	0
el	1	0	0	0	0	0
perro	0	1	0	0	0	0
Está	0	0	1	0	0	0

Tabla 2.1: Valores vectorizados Método one-hot vector

El segundo enfoque es codificar cada palabra con valor único, teniendo el mismo ejemplo anterior podemos codificarlo como, El (1) perro (2) está (3) así sucesivamente hasta finalizar la codificación, en este caso la codificación es arbitraria y el resultado sería [1,2,3,4,5,1,6], este sencillo enfoque es eficiente, pero tiene dos desventajas:

1. La codificación es arbitraria y no representa ninguna relación entre las palabras.

2. La codificación con números enteros puede ser difícil de interpretar para un modelo de clasificación lineal pues aprende un solo peso por cada palabra, esto se debe a que tampoco hay una relación entre palabras, ni similitud entre su codificación. Esta combinación entonces no sería significativa.

Finalmente el ultimo método de vectorización corresponde a embeber palabras, este método es ideal para representar palabras que podrían ser similares y esta codificación no necesita que se haga manual, este embebido corresponde a un vector de punto flotante para cada palabra y el largo de vector se puede definir antes de codificar, entonces en lugar de hacer este proceso manual hay parámetros que son entrenables (pesos aprendidos durante un entrenamiento), para esta vectorización es común ver vectores embebidos de 8 dimensiones para pequeños conjuntos de datos y hasta 1024 dimensiones para datasets muy grandes siendo esto útil capa capturar relaciones entre palabras. Un ejemplo práctico de relaciones entre palabras podría hacer referencia a la frase "yo tomo leche" esta oración es validada dentro del proceso de embebido con otras previas, entonces este método de vectorización encontraría relaciones entre "leche" y "agua" posicionándolas en algún grado de similitud cercana y en caso contrario una palabra como "perro" la posiciona lejos de alguna similitud. En este caso este proceso se realiza gracias a una red neuronal que aprende a embeber las palabras durante el entrenamiento, en principio lo hace de forma simple pero hace un backpropagation para reorganizar automáticamente.(Géron, 2017).

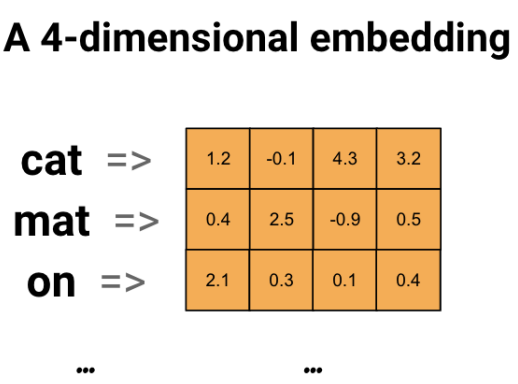


Figura 2.1: Ejemplo de Word embeddings

Todo el proceso mencionado anteriormente es llamado "Word2Vec", que es convertir las cadenas de textos a vectores, la compilación de estos documentos o frases se le denomina en procesamiento natural del lenguaje como corpus, y aquellas palabras numeradas son conocidas como léxico (Lane y col., 2019). Es posible que con este método no se pierda información sin embargo un léxico de este tipo debe tener millones de denominaciones para cada palabra, entonces si se introduce un nuevo documento de cierta variabilidad lingüística para clasificar, es posible al final en la clasificación tengamos vectores vacíos pues los recuentos de palabras podrían ser de cero.

Resumiendo, si usamos un texto en español lo primero que se debe realizar es convertir una cadena de palabras en un lista que además retire signos de puntuación, este proceso se le conoce como tokenización, enseguida lo pasamos por uno de los procesos Word2vect, anexo a esta clasificación sería buena idea eliminar palabras que no aportan variabilidad lingüística a un archivo (frase) o que tienen palabras raras del lenguaje, podríamos decir que este es un filtro y es comúnmente conocido como "stopwords".

Estos flujos de trabajo en inteligencia artificial se les suele llamar como tuberías, en donde el flujo de trabajo cambia para algunos elementos dependiendo como se programen más tareas pendientes en la tubería, el ejemplo se ve en la figura 2.2 pues el texto fluye desde la parte superior clasifica las palabras vacías o stopwords al tiempo que aísla las palabras raras cayendo el resto en la parte inferior de forma apilada en unas cajas o casillas que componen los vectores.

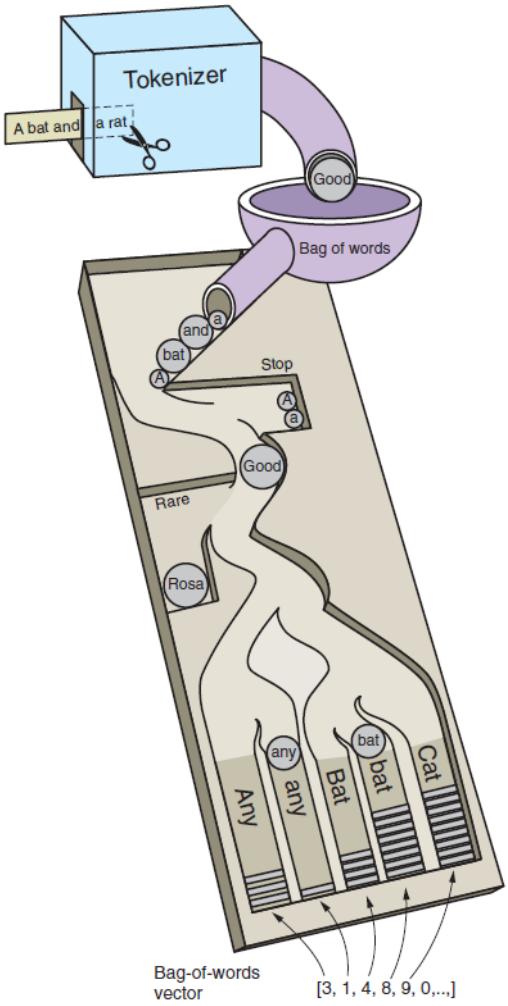


Figura 2.2: Flujo de trabajo en vectorización

Ahora que ya tenemos nuestro léxico o diccionario, el procesamiento natural del lenguaje compone vectores con un poco de perdida, no obstante, la bolsa de palabras o vectores retienen suficiente información para producir modelos de aprendizaje automático útiles e interesantes. Dentro del aprendizaje automático tenemos las máquinas de soporte vectorial, el perceptrón multicapa, las redes neuronales evolutivas y las redes neuronales recurrentes.

2.2.1. Máquinas de soporte vectorial

Las máquinas de soporte vectorial son muy útiles y versátiles en el machine learning siendo capaces de hacer clasificaciones lineales y no lineales, regresiones e incluso es capaz de identificar datos atípicos en un conjunto de valores. Este modelo es especialmente útil para clasificaciones complejas, pero de tamaño pequeño a medio.

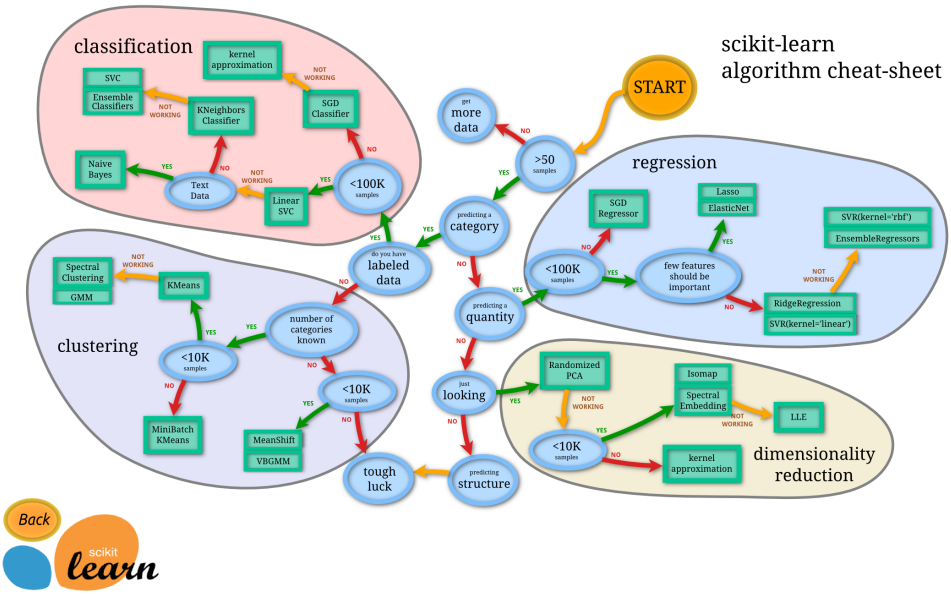


Figura 2.3: cheat sheet scikit-learn

Las máquinas de soporte vectorial se explican mucho mejor con imágenes, en la figura 2.4 se muestran dos categorías en el mismo conjunto de datos (rojo, verde), estas dos categorías pueden ser separadas con una línea recta, entonces se dice que el grupo de datos es linealmente separable. Sin embargo, para separar las categorías hay infinitos planos, el objetivo principal del algoritmo SVM es encontrar el mejor plano y límites apoyado en parámetros para separar categorías en un problema lineal o no lineal, esta línea no solo separa los datos, también busca mantenerse lo más lejos posible, es similar a trazar una vía por el medio del punto óptimo de separación y donde sus bordes sirven como margen de clasificación.

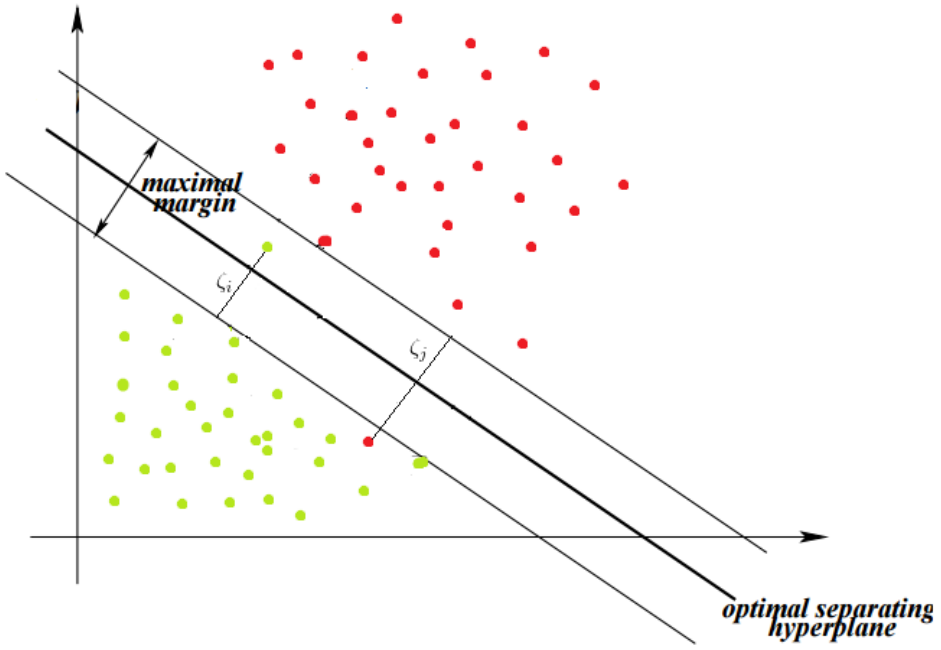


Figura 2.4: Separación de problema lineal de una máquina de soporte vectorial

2.2.2. Perceptrón Multicapa

El perceptrón multicapa permite solucionar problemas no lineales superando los problemas de su antecesor el perceptrón simple que lo relego al olvido científico (Minsky y Papert, 1969). Cuando el perceptrón fue propuesto en 1958 por Rosenblatt, Rosenblatt ya intuía el problema de clasificación en problemas no lineales y pensaba que la solución era incluir neuronas entre las capas de entrada y salida garantizando solucionar problemas no lineales e incluso más complejos. Este interrogante quedo sin respuesta durante varios años y no fue hasta la década de los 70's que paul Werbbos en su tesis doctoral propone el algoritmo Backpropagation, permitiendo el entrenamiento del perceptrón multicapa y su aplicación en la solución de variados problemas de alta complejidad.(Werbos, 1994).

En la figura se presenta la estructura del perceptrón Multicapa (MLP de sus siglas en inglés Multi Layer Perceptrón), que a diferencia del perceptrón y del Adaline, posee al menos tres niveles de neuronas, el primero es el de entrada, luego viene un nivel o capa oculta y, finalmente, el nivel o capa de salida, la función de activación normalmente suele ser sigmoidal (Caicedo y Lopez, 2017).

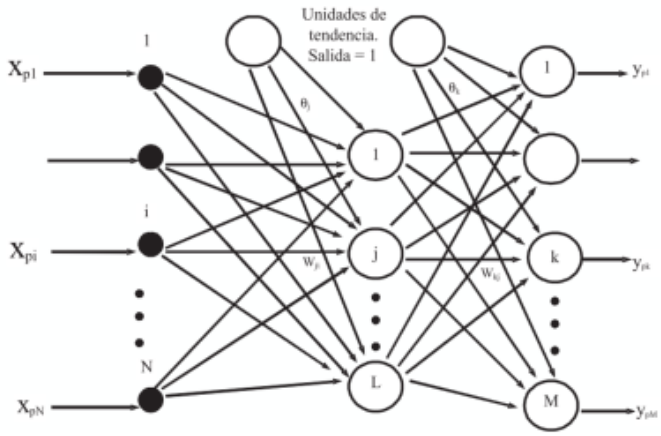


Figura 2.5: Perceptrón multicapa

Uno de los principales inconvenientes que resuelve este algoritmo es la separación de regiones que no son linealmente separables. Como el error de la capa de salida es el único que puede calcularse de forma exacta, el algoritmo propone propagar hacia atrás el error para estimar el error en las salidas de las neuronas de las capas ocultas con el fin de modificar los pesos sinápticos de estas neuronas, para explicar el algoritmo se ha decidido hacer una prueba con los datos de la imagen, con lo cual se deja claro el funcionamiento de la red neuronal.

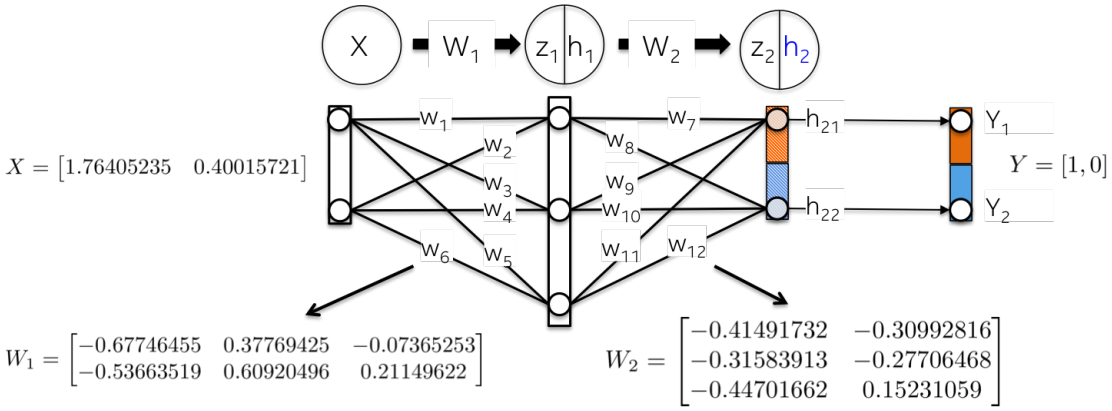


Figura 2.6: Propagación de la red hacía adelante

Los pasos para la realización del algoritmo SVM son:

1. Inicializar los pesos.

2. Mientras la condición de parada sea falsa se ejecutan los pasos 3 al 12.

3. Aplicamos un vector de entrada.

4. Calcular los valores de las entradas netas para la capa oculta.
5. Calcular la salida de la capa oculta.
6. Calcular los valores netos de entrada de la capa de salida.
7. Calcular las salidas de la red.
8. Calcular los términos del error para las unidades de salida.
9. Estimar los términos de error para las unidades ocultas.
10. Actualizar los pesos en la capa de salida.
11. Actualizar los pesos en la capa oculta.
12. Verificar si el error global cumple con la condición de finalizar.

El objetivo de la función de pérdida es cuantificar la distancia entre el vector predicho (h2) y la etiqueta proveída.

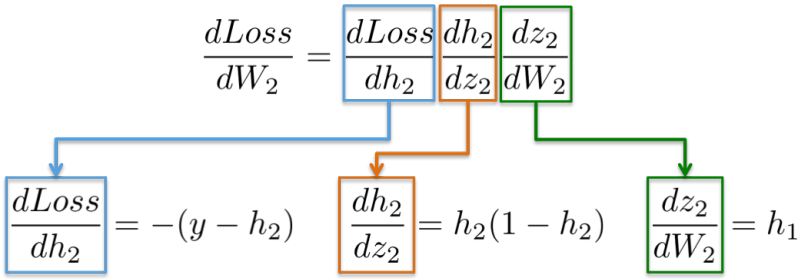


Figura 2.7: derivadas para cálculo de nuevos pesos de la red

Para actualizar los pesos en la red debemos sacar las derivadas parciales de la función de pérdida, con lo cual tenemos las ecuaciones de la imagen anterior, estas serán aplicadas y ayudarán actualizar los pesos de la red neuronal.

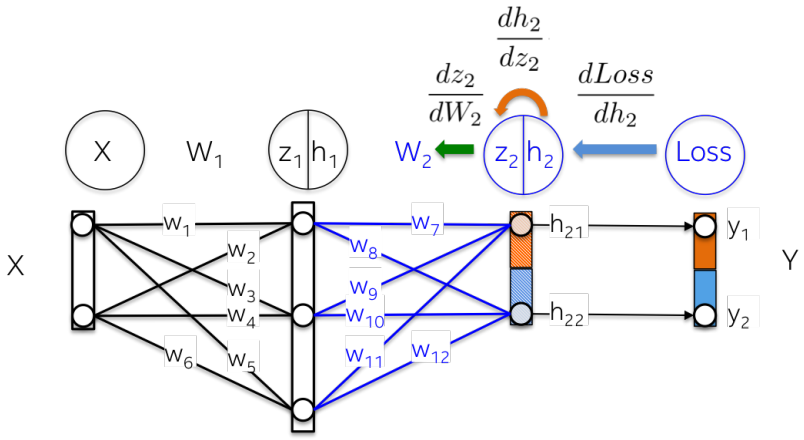
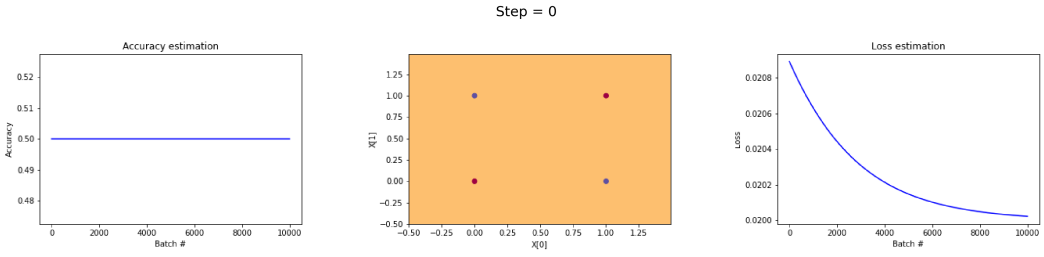
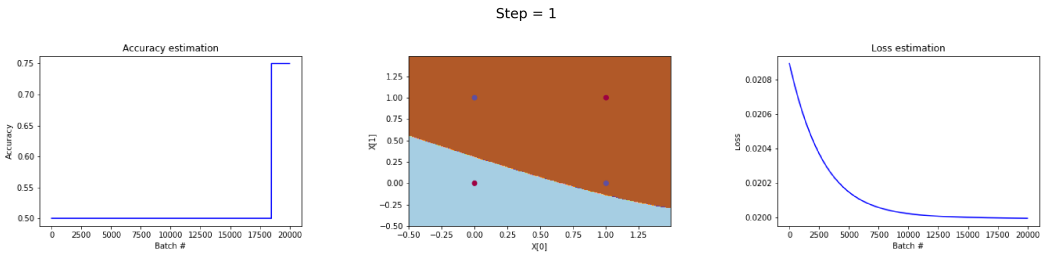


Figura 2.8: Propagación de la red hacia atrás

Aplicando las funciones podemos actualizar los pesos n veces y ahora la red está aprendiendo. Cuando se grafican los datos se pueden ver los cambios en las funciones como se muestra a continuación, buscando una correcta clasificación de una compuerta lógica.

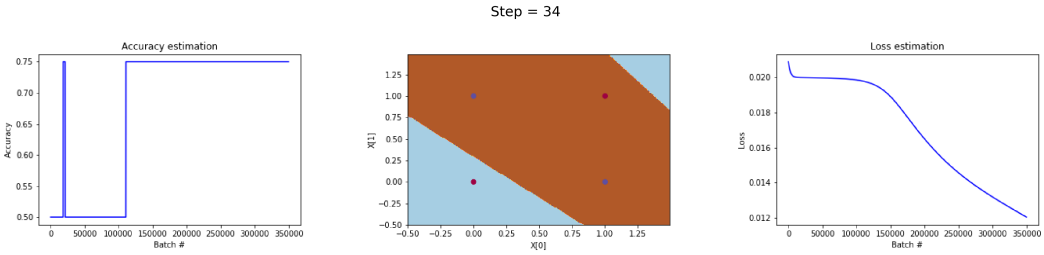


(a) Backpropagation iteración 0

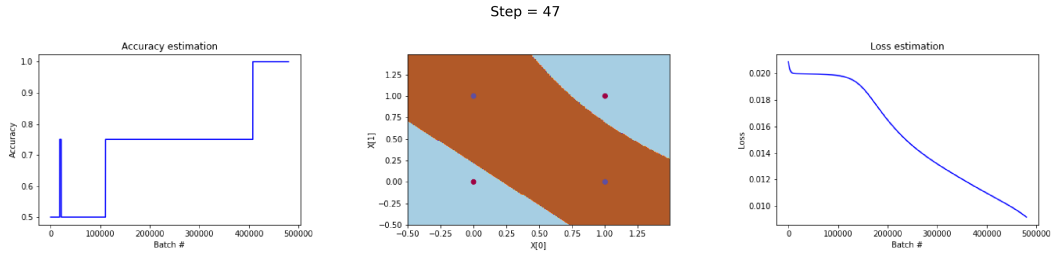


(b) Backpropagation iteración 1

Figura 2.9: pasos iniciales Convergencia de una red Backpropagation



(a) Backpropagation iteración 34



(b) Backpropagation iteración 47

Figura 2.10: Convergencia de una red Backpropagation

Las gráficas de la izquierda representan la estimación de la exactitud para la clasificación, en el medio se observa la clasificación, y en la derecha la función de pérdida, para los pasos 0, 4 y 10, se puede observar que a medida que la función de pérdida se acerca a cero la clasificación se hace más exacta y la precisión toma un carácter lineal en una mayor cantidad de épocas.

Este proceso se puede realizar fácilmente con datos de texto una vez han sido vectorizados (Word3vect), la entrada seria nuestros vectores y la salida correspondería a una categoría que le demos al texto, por ejemplo, si es una incidencia de tránsito o si el texto es la contra parte a un incidente de tránsito.

2.2.3. Redes neuronales convolucionales

Las redes convolucionales (CNN) emergen del estudio de la corteza visual del cerebro y han sido usadas en el reconocimiento de imágenes desde los 80's, en pocos años gracias al incremento en poder computacional, grandes conjuntos de datos disponibles y trucos de entrenamiento en redes profundas han conseguido desempeños increíbles en tareas visualmente complejas como carros autónomos, clasificación automática de vídeo y ha representado buenos avances reconocimiento de voz y NLP.

Las CNN reciben su nombre de un concepto extendido en matemáticas llamado convolucional, cuya tradución simple corresponde a pasar una pequeña ventana sobre un conjunto de datos.

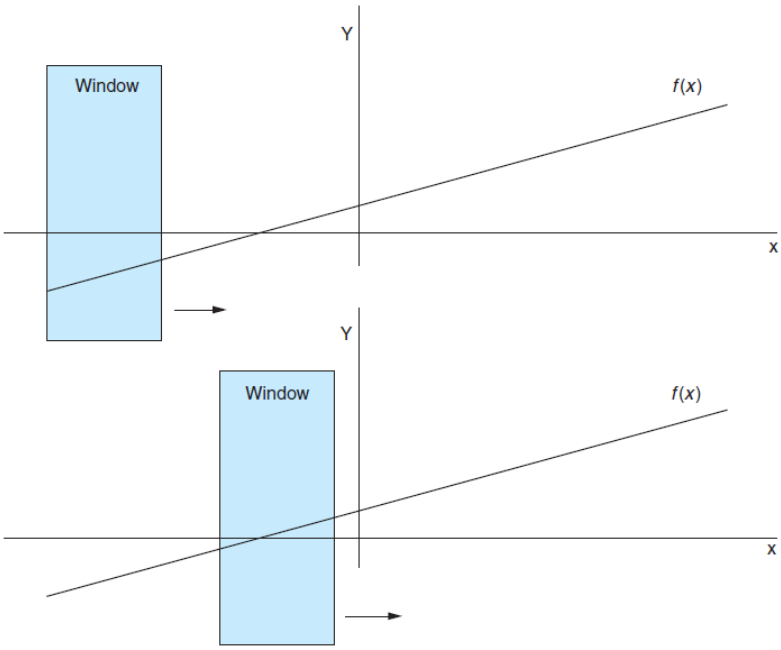


Figura 2.11: Ventana de convolución

En un primer momento estas redes toman alta importancia en procesamiento y reconocimiento de imágenes, principalmente porque la red es capaz de capturar relaciones espaciales entre píxeles vecinos y puede determinar si en una imagen hay un perro, un gato u otros objetos. La forma de hacer este proceso llamado convolución consiste en crear una máscara, kernel o ventana, una matriz cuadrada que se desliza en cada punto (X,Y) de la imagen como se muestra a continuación.

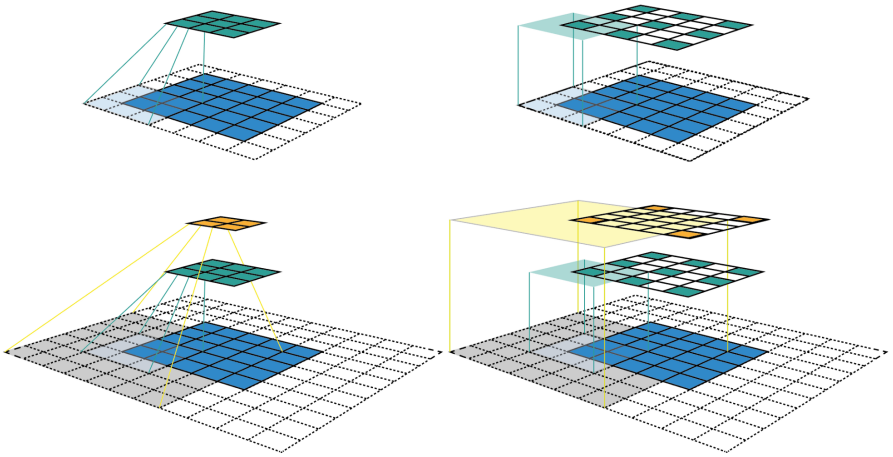


Figura 2.12: Desplazamiento del kernel sobre la imagen de entrada

El tamaño del Kernel es un parámetro que debe elegir quien construye el modelo y depende mucho del conjunto de datos, pero generalmente toma valores de matrices

de tres por tres píxeles, el movimiento de la ventana se conoce como "paso" y generalmente es de 1 píxel, si el paso fuera más grande es posible que se pierda el efecto de desenfoque con los píxeles vecinos. Este filtro o kernel por píxel de la matriz posee un conjunto de pesos y anexo igualmente una función de activación, la respuesta al mover la ventana sobre la imagen se encuentra dada por una suma de productos de los coeficientes y los correspondientes píxeles de la imagen de la zona abarcada y se pasan a la función de activación ReLU (Unidad lineal rectificada).

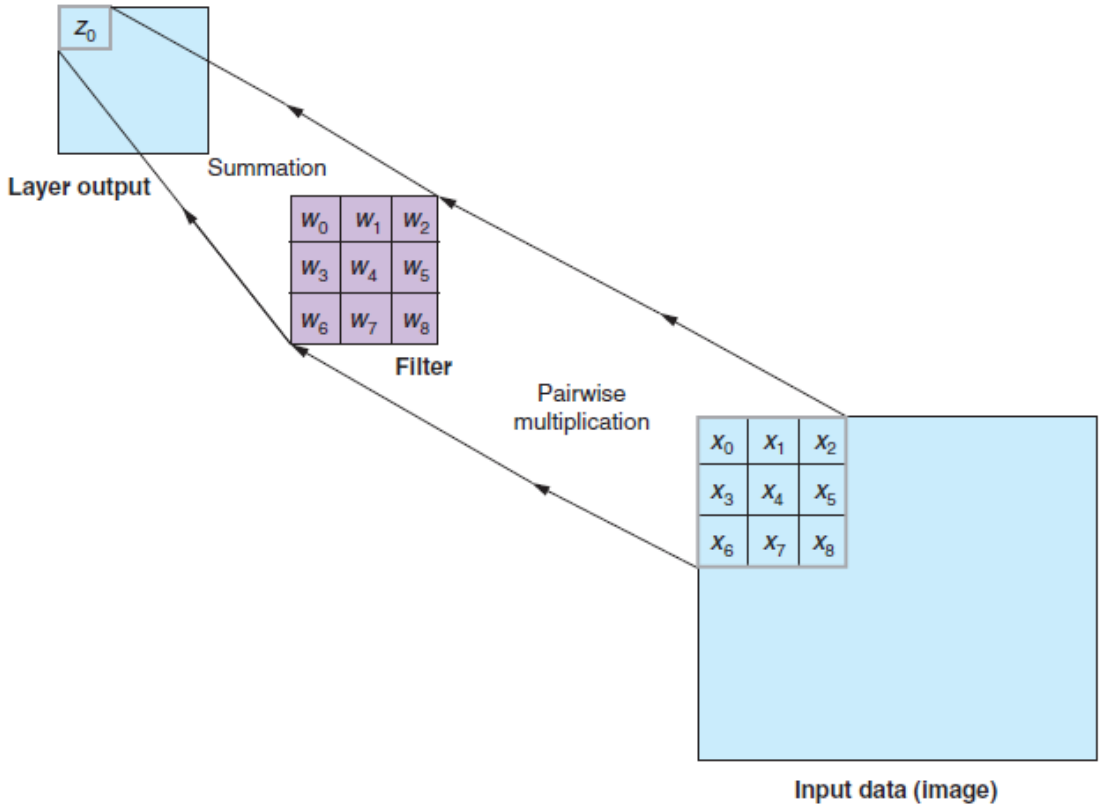


Figura 2.13: Inicialización de pesos aleatorios

Al igual que en una red neuronal los pesos se inicializan de forma aleatoria y son cercanos a cero, esto va a provocar que en las primeras iteraciones solo tengamos ruido, la clasificación tendrá muchos errores, dichos errores pueden usar la propagación hacia atrás para calcular nuevos pesos.

Lo anterior en imágenes, ¿pero y entonces como usarlo en textos?, pues resulta que se pueden utilizar redes convolucionales mediante el uso de vectores de palabras, en este caso es mejor usar el resultado del método de vectorización "Word embedding", pues como se dijo en la primera parte de este capítulo agrupa o posiciona las palabras según un grado de similaridad, en este caso como se mostró en la figura 2.1 las relaciones entre palabras de forma vertical seria arbitrarias, sin embargo, la información que si es relevante se encuentra en las posiciones horizontales.

Entonces queremos enfocarnos solo en relaciones de nuestro tokens o palabras en una dimensión espacial, en lugar de filtrar en dos dimensiones como lo hacíamos con las imágenes lo haremos en una sola dimensión lo que significa también tener un kernel de una sola dimensión como se muestra en la imagen siguiente.

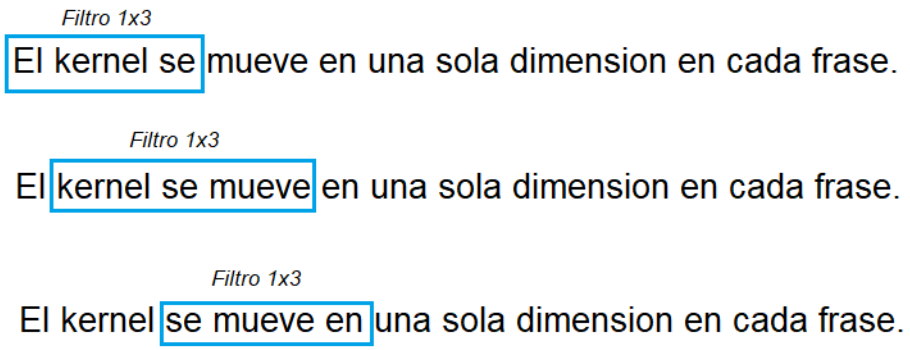


Figura 2.14: Manejo de kernel sobre un texto

En la imagen anterior se puede ver que es posible tener un kernel y moverlo por cada palabra, esto es similar a la forma en que se aplica en imágenes, ahora desagreguemos un poco más el problema para ver cómo se está aplicando la red convolucional.

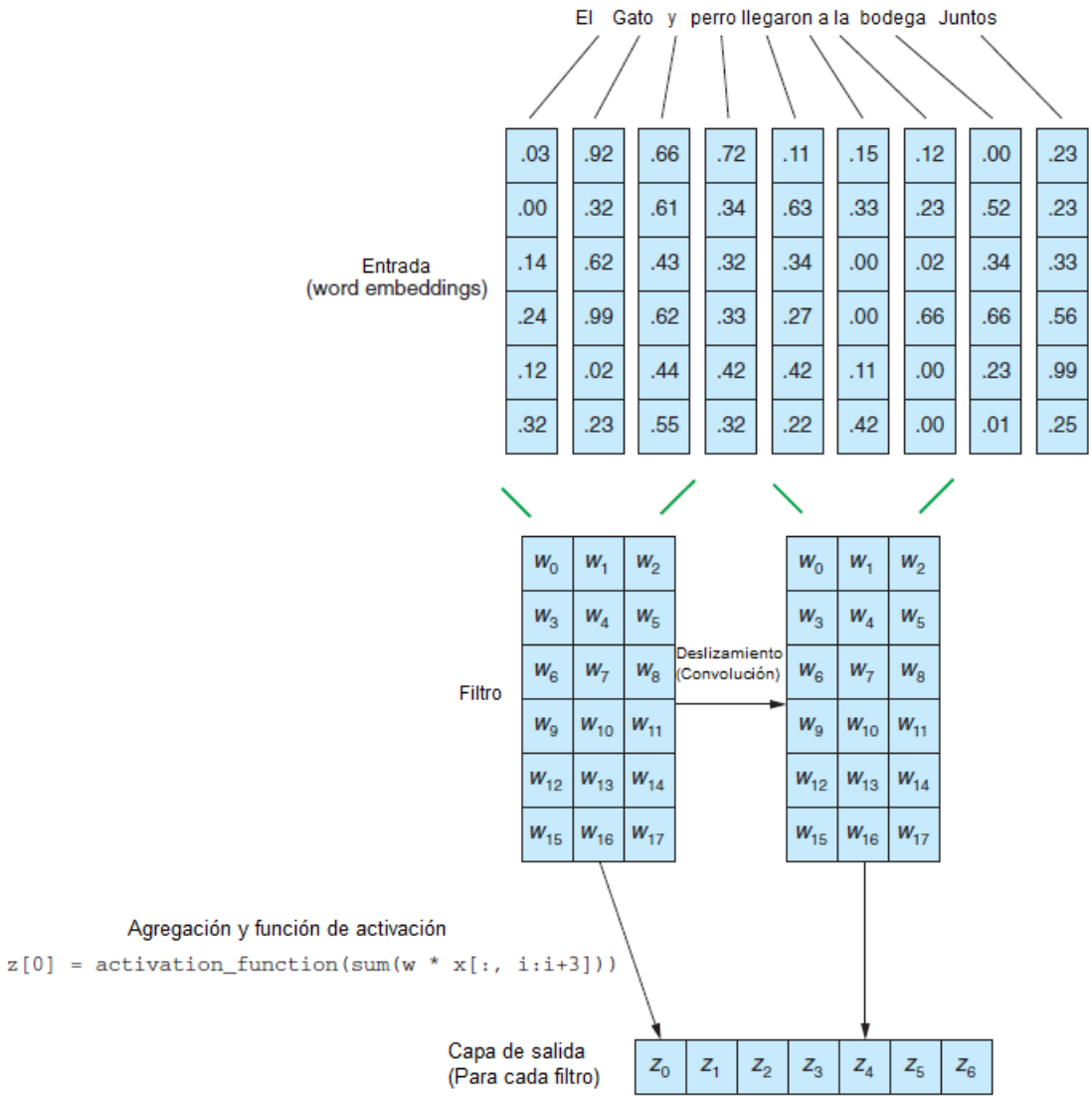


Figura 2.15: Proceso de convolución sobre texto

De la imagen anterior, la primera parte recibe nuestra oración, la cual viene anexa a una vectorización con ciertas relaciones (Word Embeddings), acá podemos imaginar estas palabras como una imagen pues con la vectorización componemos una matriz, entonces podemos pasar nuestro kernel, una matriz (3x6), con unos pesos inicializados aleatoriamente, esta ventana se desplazara por toda la "imagen" entregándonos un vector de salida, como se ha dicho esta primera iteración podría ser solo ruido, el secreto está en hacer una propagación hacia atrás para calcular nuevos pesos que ayuden en clasificaciones o identificación de elementos en nuestra arquitectura de red convolucional, de esta forma queda demostrado el potencial de este tipo de redes en el NLP. Algunas arquitecturas convolucionales que anexan el aprendizaje profundo en problemas de entendimiento del lenguaje son las siguientes:

1. BERT de Google (Devlin y col., 2018).
2. GPT de OpenAI (Peters y col., 2018).
3. GPT-2 de OpenAI (Radford y col., 2018).
4. Transformer-XL de Google (Dai y col., 2019).
5. XLNet de Google (Yang y col., 2019).
6. XLM de Facebook (Lample y Conneau, 2019).
7. RoBERTa de Facebook (Liu y col., 2019).
8. DistilBERT de HuggingFaceb (Sanh y col., 2019).
9. CTRL de Salesforce (Keskar y col., 2019).
10. CamemBERT de Inria, Facebook y Sorbonne (Martin y col., 2019).
11. ALBERT de Google Research and the Toyota Technological Institute at Chicago (Lan y col., 2019).
12. ELECTRA de Google Research y Stanford University (Clark y col., 2020).

2.3. Extracción de entidades

La extracción de entidades de un texto es algo bastante interesante, suponga que usted le dice a un asistente de búsqueda por voz, “Recuérdame buscar promociones en tienda.com el próximo lunes”, el asistente de voz debe generar una tarea en el calendario, pero antes de hacerlo tiene que saber que “recuérdame” hace referencia al usuario, una persona. También debe identificar que “tienda.com” es una url abreviada y que “lunes” hace referencia a un día de la semana y luego si asignar la tarea en el cronograma del usuario. Este problema es el que soluciona la extracción de entidades, algunas entidades que se pueden extraer son coordenadas geográficas, organizaciones, empresas, personas, entidades políticas, rangos de tiempo, eventos, fenómenos naturales, entre muchos otros. Pensemos para el mismo ejemplo un poco más sobre la complejidad de una orden, claramente además de extraer información necesita de un conocimiento base por ejemplo conocimiento almacenado en la sesión del usuario o conversaciones previas, ese conocimiento que es relevante solo en una conversación es llamado “contexto” esta información de contexto es almacenada en el conocimiento base.

Adicionalmente a las tareas de reconocimiento de entidades como se dijo en el párrafo anterior es posible que el modelo de entendimiento del lenguaje requiera un conocimiento general del mundo, en este caso no es necesario programar todo lo que una persona conoce sobre el mundo, esto lo puede aprender de oraciones sacadas de Wikipedia, por ejemplo:

"Simón Bolívar, fue un general y político venezolano, fundador de las repúblicas de la Gran Colombia y Bolivia."

Extraer conocimiento de esta frase se puede lograr mediante relaciones, reduciendo la pieza de información, esto puede almacenarse por ejemplo como:

('Simón Bolívar', 'es un', 'general')
('Simón Bolívar', 'es un', 'político venezolano')

Este ejemplo es útil para mostrar las conexiones entre las palabras y resumir información, estas relaciones generalmente se guardan en un formato estándar RDF (formato de descripción de relación) y se almacenan en archivos XLM o bases de datos, esta tripleta se compone de (Sujeto, relación, objeto) y algunas veces también son llamadas ontología lingüística que de paso permite representarse como gráfico.

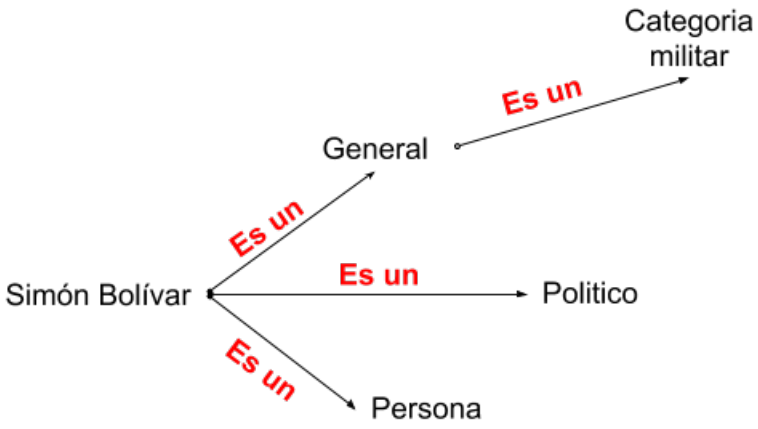


Figura 2.16: Representación gráfica modelo RDF

Este conocimiento base puede ser usado para construir sistemas inteligentes de entendimiento natural del lenguaje, en este caso convertimos información no estructurada en información estructurada o gráfica. Al contrario de lo que se pensaría en un proceso de ciencia de datos estas tareas no son solo aprendizaje no supervisado,

esta lógica sobre cada una de las palabras debe ser construida por medio de patrones regulares, para evitar al máximo la intervención humana.

Las expresiones regulares son cadenas de texto escritas en un lenguaje de programación especificando un algoritmo de búsqueda para una expresión o cadena de caracteres en específico. Estas expresiones regulares son muy útiles y concisas e intentan coincidir en algún punto con un input que entra a un algoritmo construido, por ejemplo si alguien quiere recuperar coordenadas geográficas de un texto podría programar un algoritmo que haga uso de expresiones regulares y que de paso valide si es o no una coordenada, un ejemplo de esto en lenguaje de programación Python se puede ver a continuación.

```
import re
lat = r'([-]?[0-9]?[0-9][.][0-9]{2,10})'
lon = r'([-]?1?[0-9]?[0-9][.][0-9]{2,10})'
ep = r'[,/ ]{1,3}'
re_gps = re.compile(lat + sep + lon)
re_gps.findall('http://...maps/@34.0551066,-118.2496763...')
# respuesta!
[(34.0551066, -118.2496763)]
```

Figura 2.17: Implementación simple, extracción de coordenadas geográficas

Como se puede ver, lat y lon hacen uso de expresiones regulares, si compilamos la expresión e intentamos hacer una búsqueda sobre un texto tendremos que recupera información en forma de una lista de datos.

2.4. Recuperación de entidades geográficas

Los sistemas de recuperación de información espacial (GIR) o foco geográfico, buscan un lugar o lugares en los que se centra el contenido de un texto, son sistemas especializados de los sistemas tradicionales de recuperación de información enfocados a la obtención de documentos relevantes para una determinada localización (Peregrino y col., 2013), es común encontrar trabajos que una vez hacen reconocimiento de entidades nombradas y contexto geográfico intentan una posterior desambiguación de topónimos, a fin de concretar cual entre múltiples localizaciones se puede asociar un texto, por ejemplo en Colombia todas las ciudades cuentan con una plaza o parque principal con nombre “Simón Bolívar”, por lo cual se debe discriminar a cual lugar en el espacio hace referencia una cadena de texto, esto se puede ver en (Leidner, 2008), (Martins y Silva, 2005) y (Anastácio y col., 2009), otros investigadores han

afrontado el reto de clasificación y desambiguación con ayuda de diferentes corpus de datos como GeoNames para construir frameworks escalables como Geotxt (Karimzadeh y col., 2019) y e implementaciones NER prediseñadas en diferentes librerías, concluyendo La recuperación de información geográfica se apoya en la extracción de entidades sin embargo el problema de la desambiguación sigue estando presente por lo que se han buscado nuevas formas de abordar este problema (Purves y col., 2018).

2.5. Estado del arte

El problema de flujos de tráfico se ha estudiado desde hace décadas, en matemática discreta se puede plantear como el camino más corto entre dos nodos de un grafo dirigido o no dirigido, con la evolución de los SIG, este problema se extrapola a la cotidianidad del transporte, donde las aristas son las calles con flujo y los nodos aquellos sitios geográficos donde se cortan las aristas, en la ciencia de la información geográfica estos problemas se abordan desde el área de análisis espacial donde se pretende con geoestadística y geo procesos dar explicación o probabilidad de recurrencia de un evento en el espacio, sin embargo estas técnicas no son suficientes cuando la cantidad de datos disponible supera las capacidades de los Sistemas de información geográfica, por ejemplo en mayo de 2004 el problema del vendedor ambulante (TSP) se resolvió con 24.978 ciudades suecas, al final se obtuvo una ruta de 72.500 Km de longitud comprobándose la mejor ruta, con esto se convirtió en el problema TSP más grande solucionado hasta la actualidad superando la solución de 15.112 ciudades del 2001, dicho cálculo fue posible con un clúster de 96 estaciones de cómputo Intel Xeon de doble procesador en Georgia tech, sin este clúster el tiempo en máquina de un intel Xeon de 2.8 GHz hubiese sido de 84.8 años (Applegate y col., 2006), entonces queda demostrado que a mayor cantidad de datos debe existir un aumento en poder de cómputo y algoritmia, este siendo un cálculo de ruta más corta, que pasaría si asignáramos otros valores como velocidades, localización de eventos públicos o choques en la vía, la complejidad subiría y es posible que no podamos cuantificar el peso que ejerce cada variable sobre la red, pensando en esta complejidad y viendo que es imposible tener un monitoreo metro a metro se plantea explotar los datos disponibles para identificar escenarios de tráfico o accidentalidad en una ciudad.

Entonces los datos recolectados son útiles si se explotan con nuevas técnicas de análisis, en el caso de la movilidad urbana, es posible obtener zonas más concurridas a ciertas horas del día, ubicaciones estratégicas según la población flotante, incremento

de fuerza de venta en zonas, valores del suelo según localización e incluso distribución inteligente de negocio, todos estos se pueden comportar como patrones espaciales afectando el desarrollo sostenible de una sociedad, entonces el uso de datos agregados aporta ideas para encaminar una ciudad inteligente.

El diseño de soluciones de movilidad apoyado en técnicas de análisis de datos que permiten identificar patrones en zonas urbanas que no son claramente identificables con análisis convencionales, plantean un nuevo escenario de decisiones apoyadas en Big Data con posibilidad de generar algoritmos inteligentes que dinámicamente pueden dar una probabilidad a un escenario de movilidad lenta en un sector dependiendo de la dinámica de variables en el tiempo, por ejemplo accidentes, protestas, eventos públicos, cierres viales, estado de vía, ocupación de parqueaderos, entre otros.

2.5.1. Recolección de datos de tráfico urbano

Existen varias formas de capturar información relacionada con las redes de transporte, dentro de estos se encuentran, los sistemas de tráfico activo que controla una variable clave que es la del límite de velocidad, es usada comúnmente para mejorar el tráfico y reducir choques de automotores, otro tipo de información es la registradas por las entidades policiales de tránsito que reportan a una base de datos generando un histórico con localización, tipo de involucrados, gravedad, hora y fecha, combinando bases de datos espaciales. Sin embargo, la efectividad de la detección tradicional de incidentes a menudo está limitada por la escasa cobertura de sensores sobre una malla vial y la notificación de incidentes a los sistemas de respuesta requieren de mucha mano de obra (Gu y col., 2016).

Como respuesta y para detectar incidentes de forma rentable en tiempo real, los investigadores han usado técnicas y métodos diversos, entre estos se encuentran; Método de California, Algoritmos estadísticos discriminativos, Máquinas de Soporte Vectorial, Red neuronal artificial, Filtro Kalman, Sequential Probability Ratio Test (SPRT), Modelo auto regresivo de promedio móvil integrado (ARIMA), Algoritmo de alta ocupación (HIOCC), modelos wavelet y Agrupaciones. Otros investigadores pensando en la rentabilidad del acceso a datos en tiempo real, han evaluado el crowdsourcing como alternativa a la recolección de datos, concluyendo que estos datos tienen capacidad para trabajar en tiempo real y además tienen precisión razonable.

Otras formas de identificar patrones de tráfico se han implementado más recientemente, una de estas técnicas es el modelamiento estadístico tradicional, como la regresión logística bayesiana de la cual los investigadores asumieron una relación lineal entre variables dependientes e independientes, otra técnica empleada en

la identificación de patrones de tráfico son las máquinas de soporte vectorial que se basa en la teoría de aprendizaje estadístico, una tercera técnica es la empleada con redes neuronales, que aunque unos autores indican que son cajas negras (Yu y Abdel-Aty, 2013) otros les sacan provecho con aprendizaje profundo.

Finalmente, entusiastas han minado las redes sociales, dando como resultado investigaciones que presentan metodologías para rastrear, procesar y filtrar tweets públicos en tiempo real, luego se analizan para extraer información sobre incidentes de tránsito a través de aprendizaje profundo (Zhang y col., 2018).

2.5.2. Extracción de información

En la revisión de bibliografía sobre algoritmos de detección de incidentes tradicionales hay dos categorías, la primera la adquisición de los datos y la segunda el análisis de datos, los datos alimentados a un detector se clasifican en dos categorías:

1. captura fija: conteos, ocupación y velocidades medidas por sensores inductivos, magnéticos, microondas, infrarrojo, ultrasonido acústico, láser o procesadores de vídeo.

2. Datos de vehículo: trayectoria de vehículo muestreada a partir de GPS , geolocalización satelital o identificación automática.

Dependiendo del sistema de adquisición de datos se han desarrollado diferentes investigaciones entre estas están; la identificación de patrones en el flujo de estructuras viales en eventos de incidente con datos de sensores de captura fija (Stephanedes y Chassiakos, 1993), este tipo de investigaciones en detección de incidentes en una red vial controlada por señales se remontan al método California (Thancanamootoo, 1988), algoritmos como este hacen la suposición de que la tasa de flujo, dada la ocurrencia de un incidente, se reduce y la ocupación de la vía asciende (H. J. Payne y S. C. Tignor, 1978).

Otras formas de identificar patrones de tráfico se han implementado más recientemente, una de estas técnicas es el modelamiento estadístico tradicional, como la regresión logística vayesiana de la cual los investigadores asumieron una relación lineal entre variables dependientes e independientes, otra técnica empleada en la identificación de patrones de tráfico son las máquinas de soporte vectorial que se basa en la teoría de aprendizaje estadístico, una tercera técnica es la empleada con redes neuronales, que aunque unos autores indican que son cajas negras (Yu y Abdel-Aty, 2013) otros les sacan provecho con aprendizaje profundo

Aprovechando la era de la información y la explotación de datos de la web, entusiastas han minado las redes sociales, entre estas específicamente Twitter, dando como resultado investigaciones que presentan metodologías para rastrear, procesar y filtrar tweets públicos en tiempo real, luego se analizan para extraer información sobre incidentes de tránsito a través de un procedimiento simple usando técnicas de procesamiento de lenguaje natural, clasificación y georreferenciación de textos que reportan a la red social. (Gu y col., 2016), (Zhang y col., 2018).

Otras investigaciones han abordado el tema de identificación de patrones desde la parte geográfica, más específicamente desde el análisis espacial y los sistemas de información geográfica, en este caso se construyen correlaciones espaciales sobre áreas con alta probabilidad de ocurrencia de incidencias viales, cuantificándose con datos pasados para toma de decisiones basada en el conocimiento del riesgo de incidencias que afecten el tráfico urbano, dentro de estas están estudios como los realizados en Bogotá por (Vargas y col., 2012) y metodologías como la propuesta por la agencia nacional de seguridad vial (Agencia Nacional de seguridad Vial, 2012).

Capítulo 3

CLASIFICACIÓN Y SEGMENTACIÓN

En este capítulo se muestra el proceso de recolección de información de Twitter, los tweets de los usuarios y las reglas de uso de los datos, igualmente se muestra los atributos que se pueden extraer ya sea del usuario o los mensajes públicos de estado, a la vez que se crean filtros de búsqueda en el API de desarrollo orientados a la problemática planteada en esta investigación.

Dado que la captura de datos se realiza de forma automatizada se plantea un modelo de recolección y uno de almacenamiento pues en este caso tratar con formato de textos, como 'txt' no es viable por la cantidad de información recolectada. Se plantea una base de datos Postgres que contiene varias tablas como lo son los datos iniciales de entrenamiento, validación, y no menos importante los datos a clasificar en donde se identifica si el contexto del tweet reporta alguna incidencia de tipo vial.

La parte final de este capítulo se ocupa de la extracción del contexto geográfico sobre aquellos textos que el modelo de inteligencia artificial decidió que eran incidencias de tipo vial, dicho contexto geográfico y dada la escala de trabajo para la ciudad de Bogotá se soluciona la extracción mediante la nomenclatura vial incrustada en el texto o lugares geográficos mencionados, con este contexto geográfico formateado correctamente se hace uso de un API de georreferenciación especializada para Bogotá que al hacer los 'request' entrega información como; LON, LAT, UPZ, Nombre catastral, dirección ingresada, dirección traducida, entre otras.

3.1. Captura de datos

En búsqueda de la mejor plataforma de recolección de información de texto no estructurado se evalúo principalmente Twitter por su gran facultad para describir el mundo en tiempo real, Twitter es denominada una red social de micro blogging que permite a sus usuarios actualizar sus estados, esto históricamente se limitó a mensajes de 140 caracteres, pero que a través del tiempo se ha ampliado y es posible que siga cambiando en el futuro. Otra buena característica de twitter útil en esta investigación es su diferencia principal con Facebook o LinkedIn, en donde los usuarios deben aceptar quien los sigue y puede acceder a su información, esto no pasa en twitter, un usuario puede ver los mensajes de cualquiera sin restricciones o seguirlo para tener los mensajes en el "time line". La esencia de twitter son los tweets que evidentemente contienen conocimiento en cadenas de texto de un determinado usuario, pero realmente guarda muchos más metadatos de lo que se pensaría, cada tweet tiene anexo dos piezas importantes de metadatos que particularmente son llamadas: entidades y lugares. Las entidades son menciones a otros usuarios, etiquetas, Urls y contenido multimedia (fotos o vídeos) que pueden asociarse al tweet, y los lugares en un tweet hace referencia a localizaciones en el mundo real, teniendo en cuenta que un lugar puede ser definido por las coordenadas o una referencia en el texto.

Afortunadamente los tweets y metadatos pueden ser accedidos por medio de una API (Application Programming Interface) liberada por la compañía Twitter (Twitter Inc., 2020) [42], solo hace falta activar el servicio y obtener las claves de acceso, la autenticación debe hacerse con ayuda de algún lenguaje de programación soportado por la APIREST. Una vez autenticado ante el servicio las opciones de consulta varían desde temas en tendencia, tweets usuarios, hasta búsquedas por palabras o hashtags. Es importante mencionar que cada tweet recuperado se encuentra en formato JSON (JavaScript object Notation) y tiene información muy variada adicional al texto del tweet alguna información adicional corresponde a: identificador, fecha del tweet, datos del usuario (nombre, nickname, foto de perfil, etc), texto, numero de retweets, localización, entre muchos otros. Es de considerar que existen límites de uso sobre el servicio y en caso de romper dichos límites es posible adquirir una nueva tasa de uso en una versión de pago de la API, también es muy importante las políticas internas de la red social no es común encontrar tweets con coordenadas pues por defecto es una función que no está habilitada para recuperarse con la API y aquellas excepciones son autorizadas por el usuario, es decir, es información anonimizada, lo mismo sucede con contenido multimedia pues como es bien sabido por ejemplo las fotos tomadas con dispositivos móviles suelen guardar metadatos entre estos coordenadas geográficas del sitio de toma, la política de twitter sobre

estos archivos es eliminar totalmente los metadatos para salvaguardar la seguridad y privacidad de los usuarios.

3.2. Filtros de captura

Uno de los aspectos comunes que se realizan cuando se trabaja con información de redes sociales es medir los temas en tendencia, en este caso podemos filtrar aquellos textos que contienen etiquetas e identifican temas relevantes al momento de la captura, esta búsqueda se hace como un 'query' dentro del API, en este caso es muy sencillo pues se busca sobre las entidades de tipo etiqueta, además un usuario del servicio puede obtener el texto original del tweet o incluso saber si alguien le ha hecho retweet. A partir de estos datos también se puede capturar aquellos tweets más virales en la web, estos son poderosas herramientas para la recolección de tendencias e interpretación de hechos, sin embargo, una de las características más importante para este trabajo es la posibilidad de extraer textos que contengan términos importantes para esta investigación, en este caso se ha iniciado la búsqueda con una semilla de palabras por ejemplo la palabra 'choqué', 'accidente','varado' entre otras y se han recolectado los resultados, estos fueron analizados y se evaluó la variabilidad lingüística de los mensajes, adicionalmente se identificaron cuentas distritales encargadas de comunicar eventualidades en la red vial de la ciudad de Bogotá. En conclusión, los textos de la semilla y los perfiles fueron consolidados y se obtuvieron las frecuencias de las palabras obteniendo un conjunto bien definido de aquellas relevantes a la hora de informar una eventualidad en las vías. Las palabras con mayor frecuencia fueron separadas y tomadas como la semilla de captura definitiva para esta investigación.

3.3. Definición Modelo de almacenamiento

Dado que se espera capturar altas cantidades de datos, lo mejor para la investigación era implementar un modelo de almacenamiento que permitiera el uso de información alfanumérica desde el principio y que hacia el final lograra almacenar información espacial de ser necesario, en este caso la mejor solución es usar una base de datos como PostgreSQL anexando su extensión PostGIS que permite gestionar objetos geográficos [43], de forma que añade la capacidad de usar PostgreSQL como una base de datos espacial que puede ser consultada por distintas vías, la más común es por medio de sistemas de información geográfica (QGIS, gvSIG, Udig, OpenJUMP) pero al tratarse de altas cantidades de datos a veces las aplicaciones convencionales

presentan bajo rendimiento. La segunda opción es usar las funciones internas de PostGIS con la limitación que no cuenta con un ambiente gráfico directo, otra posibilidad de consulta es usar un lenguaje de programación con librerías especialmente desarrolladas para tratar altos volúmenes de información espacial, en el caso de esta investigación el lenguaje de programación Python con la librería optimizada para datos espaciales GeoPandas [44] que cuenta con la posibilidad de ejecutarlo en un servicio interactivo de computación como Jupyter y sus notebooks, estos notebooks muestran un ambiente gráfico inmediato de los resultados con librerías como fiona, gmaps, plotly, PYSAL, entre muchas otras.

En total se plantearon 4 tablas de almacenamiento. La primera y segunda tabla se trabajaron con la semilla de palabras consolidada, es decir cada uno de los tweets recuperados contiene una o más palabras que pertenecen a la lista consolidada, la primera agrupa los datos iniciales de entrenamiento que se clasificaron manualmente, la segunda recopila los datos del grupo de control que también son clasificados de forma manual, estos son necesarios después del entrenamiento del SVM, y ayudan a evaluar cuál es el porcentaje de falsos positivos o falsos negativos de la matriz de confusión, básicamente muestran que tan bueno es el modelo planteado para clasificar información. La tercera tabla en la base de datos se llama “tweets” tiene como función recopilar gran cantidad de datos que el modelo no ha evaluado ni en su entrenamiento ni validación del entrenamiento tampoco los datos han sido clasificados como en las dos tablas descritas anteriormente, por lo que la idea de esta tabla es pasarla por el modelo ya entrenado y validado para su clasificación para posteriormente separar las incidencias viales identificadas hacia una tabla especial que se denominó “incidencias”, donde además contener incidencias se han agregado nuevos campos que son el output del algoritmo de extracción de contexto geográfico, es decir, en estos campos se almacenan localidad, unidad de planeación (UPZ), latitud, longitud y un quinto valor denominado ’success georef’ que se encarga de verificar si la obtención de coordenadas fue posible.

El resumen de la estructura de las tablas se muestra a continuación.

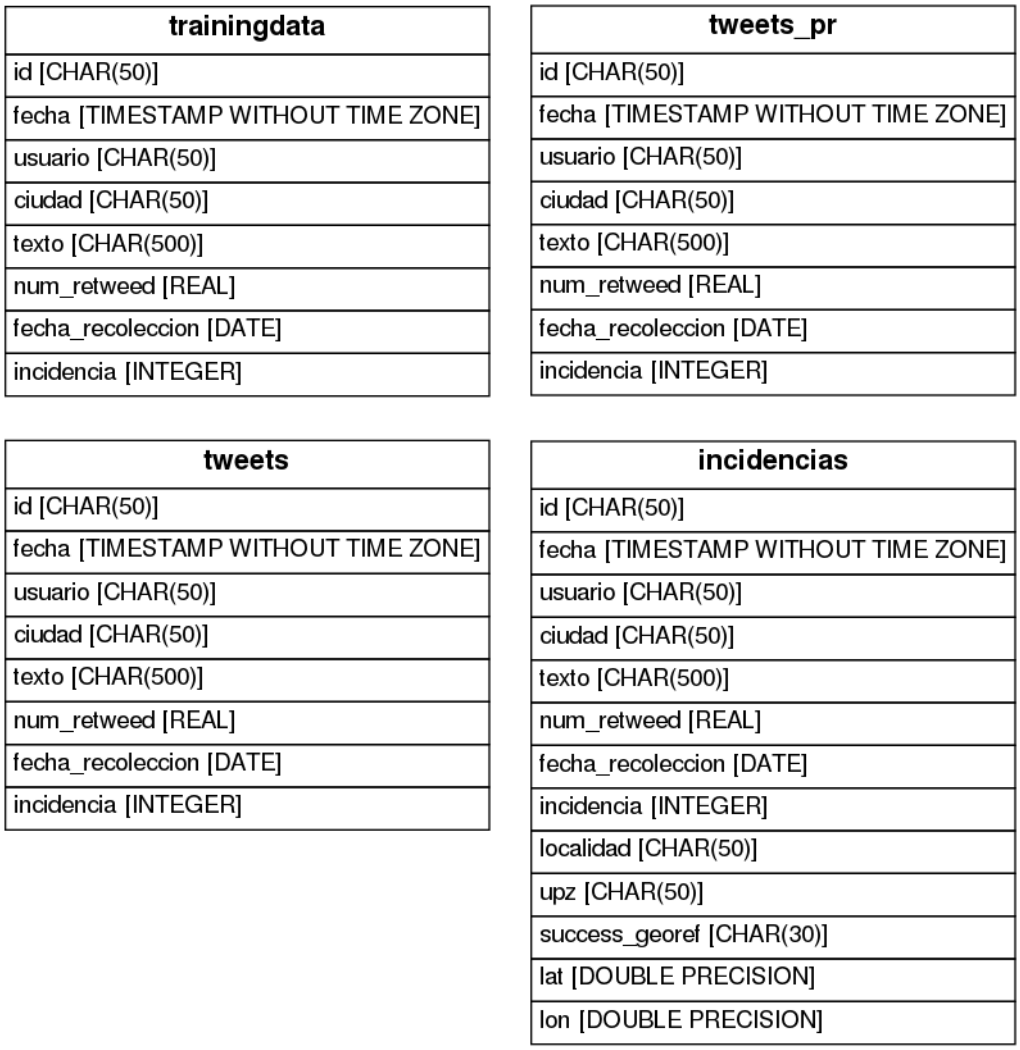


Figura 3.1: Estructura detallada de tablas

3.4. Tweets de incidencias de tráfico

Como se ha explicado anteriormente es posible generar variedad de filtros, es esencial para esta investigación generar un filtro de zona, esto quiere decir que todos los resultados deben pertenecer a la ciudad de Bogotá con algunos municipios vecinos, esto se logra mediante un 'query' pasando un único par de coordenadas de la ciudad y un radio de búsqueda, este filtro pareciera ser un filtro espacial, sin embargo es la metadata de los usuarios la que se compara, por ejemplo un método consiste en monitorizar el último lugar donde se inicia y cierra una sesión dentro de la aplicación, la forma de ubicar los terminales de hardware es variado, la ubicación de un usuario

se puede determinar de tres formas, la primera es el GPS del celular en aquellos en que tiene permiso de activarlo, la segunda se realiza mediante la IP del terminal y una última es la triangulación de antenas de telecomunicaciones.

El segundo paso importante es la recolección de textos a partir de un conjunto definido de palabras, anteriormente se denominó semilla, esta semilla se obtiene de usuarios específicos y según su frecuencia, las representativas fueron las siguientes:

Semilla = ('transito', 'varado', 'choque', 'accidente', 'incidente', 'vial', 'siniestro', 'particular', 'ciclista', 'motociclista','ambulancia','bomberosbogota', 'bomberos', 'motocicleta', 'peatón', 'criminalista', 'conduciendo', 'embriaguez', 'manifestación', 'marcha')

Veinte palabras tuvieron como evidencia mayor frecuencia, cada palabra retorna 100 resultados (tweets), en total por cada vez que se corre el hilo de captura tenemos un total de 2000 tweets, el máximo que retorna la cuota de acceso para una cuenta gratuita de la API, en el caso que se tengan mayor cantidad de palabras relevantes se debe acceder a la capa de pago de la API.

Cosas adicionales que debe tener en cuenta el código de acceso y registro en la base de datos es que Twitter generalmente devuelve los mismos valores cuando se hacen solicitudes en corto tiempo, es obligatorio identificar el ID del tweet y compararlo con los registros de la base de datos, en caso de encontrar el mismo identificador lo salta y omite registrarlo nuevamente, también se debe tener en cuenta en la metadata si es un retweet pues aunque el identificador es diferente la información de texto del tweet es corresponde al original, muchas veces ya capturado, en este caso el algoritmo debe identificar si un usuario publicó un mensaje de otro para evitar redundancia en la información.

Con lo anterior claro se generan las funciones que permiten realizar un flujo de trabajo, cada vez que se ejecuta la función principal debe permitir capturar tweets, validar su originalidad, evitar guardar información idéntica e iterar sobre cada palabra de la semilla. Para evitar ejecutar el código manualmente cada vez que se quieren nuevos datos es ideal tener una programación de la tarea de captura, esto en Linux se realiza con Crontab, este comando permite ejecutar la función principal cada cierto tiempo, en el caso de esta investigación la captura se realiza cada media hora todos los días de la semana.

3.5. Clasificación de información de tránsito

El proceso de clasificación se enfoca en dos conjuntos de datos, el primero se encuentra subdividido en el grupo de entrenamiento y validación, con una proporción 70:30 respectivamente y seleccionados de forma aleatoria, este conjunto de datos es clasificado manualmente con valores de 1 y 0, el número 1 identifica incidencias de tránsito y el 0 aquellos que no tienen nada que ver con incidencias, este conjunto de datos está compuesto por 2.432 tweets, de los cuales 1.012 resultaron ser reportes relacionados con tránsito y el restante, 1.420, no hacen referencia al tema de interés. El segundo conjunto se capturó con los respectivos filtros mencionados en las secciones anteriores un total de 69.274 tweets componen este conjunto, la ventana de tiempo en los que se ha capturado la información va desde el 9 de septiembre de 2019 al 20 de julio de 2020, es decir aproximadamente 11 meses de captura de información.

3.5.1. Entrenamiento Máquinas de soporte vectorial y afinado del modelo

Como se dijo anteriormente se clasificó un conjunto de datos de forma manual que sirve a las tareas de entrenamiento y validación, sin embargo antes de pasar al entrenamiento la información debe ser pre procesada para representarla como vectores, las tareas realizadas corresponden a las siguientes:

A Tokenizar, convierte la cadena de texto en una lista de palabras y remueve signos de puntuación.

B Convertir todas las palabras en minúsculas.

C Remover “stopwords”, las “stopwords” son palabras frecuentes que no le aportan valor a las frases, algunos ejemplos serían las palabras “de”, “por”, “con”, entre otras.

D Stemming, es el proceso por el cual se transforman palabras a su raíz, ejemplo “maravilloso”, “maravilla” o “maravillarse”, en este caso se considerarán la misma palabra.

Una vez la información ha sido vectorizada es normal en cualquier algoritmo de machine learning encontrar los mejores parámetros que posibiliten la clasificación cometiendo la menor cantidad de errores, en este caso se va a correr varias veces un

```
parameters = {
    'vect__max_df': (0.5, 1.9),
    'vect__min_df': (10, 20,50),
    'vect__max_features': (500, 1000),
    'vect__ngram_range': ((1, 1), (1, 2)),
    'cls__C': (0.2, 0.5, 0.7),
    'cls__loss': ('hinge', 'squared_hinge'),
    'cls__max_iter': (500, 1000)
}
```

Figura 3.2: Rango de hiper parámetros para el entrenamiento óptimo

proceso de entrenamiento y prueba para finalmente ponerlos en un ranking de las mejores soluciones, el espectro de parámetros en el que se va a buscar es el siguiente:

La combinatoria de estos parámetros dan como resultado 288 experimentos donde cada experimento usa una combinación diferente de parámetros dividiendo el grupo de datos en entrenamiento y validación en un ciclo donde n=4, es decir, prueba el modelo n veces y toma el puntaje obtenido para continuar al siguiente experimento. Algunos resultados fueron:

Tabla 3.1: Muestra de ranking de experimentos variando parámetros del modelo

Rank	**Parámetros**	**split0**	**split1**	**split2**	**split3**	**split4**	**Media Test**	**Desviación Test**
4	{'cls__C': 0.2, 'cls__loss': 'hinge', 'cls__max_iter': 500, 'vect__max_df': 1.9, 'vect__max_features': 500, 'vect__min_df': 10, 'vect__ngram_range': (1, 1)}	0,976	0,973	0,958	0,965	0,917	0,958	0,021
49	{'cls__C': 0.2, 'cls__loss': 'squared_hinge', 'cls__max_iter': 500, 'vect__max_df': 1.9, 'vect__max_features': 500, 'vect__min_df': 50, 'vect__ngram_range': (1, 1)}	0,971	0,970	0,972	0,968	0,889	0,954	0,032
117	{'cls__C': 0.5, 'cls__loss': 'squared_hinge', 'cls__max_iter': 500, 'vect__max_df': 0.5, 'vect__max_features': 1000, 'vect__min_df': 50, 'vect__ngram_range': (1, 1)}	0,970	0,969	0,970	0,968	0,886	0,953	0,033
179	{'cls__C': 0.5, 'cls__loss': 'squared_hinge', 'cls__max_iter': 500, 'vect__max_df': 1.9, 'vect__max_features': 1000, 'vect__min_df': 10, 'vect__ngram_range': (1, 2)}	0,968	0,969	0,957	0,955	0,908	0,951	0,022

Continua en la siguiente pagina

Tabla 3.1 – *Continuación de la tabla anterior*

Rank	**Parámetros**	**split0**	**split1**	**split2**	**split3**	**split4**	**Media Test**	**Desviación Test**
179	{'cls__C': 0.5, 'cls__loss': 'squared_hinge', 'cls__max_iter': 1000, 'vect__max_df': 0.5, 'vect__max_features': 1000, 'vect__min_df': 10, 'vect__ngram_range': (1, 2)}	0,968	0,969	0,957	0,955	0,908	0,951	0,022
244	{'cls__C': 0.5, 'cls__loss': 'hinge', 'cls__max_iter': 500, 'vect__max_df': 0.5, 'vect__max_features': 500, 'vect__min_df': 10, 'vect__ngram_range': (1, 2)}	0,968	0,968	0,943	0,958	0,895	0,947	0,027
269	{'cls__C': 0.5, 'cls__loss': 'squared_hinge', 'cls__max_iter': 1000, 'vect__max_df': 1.9, 'vect__max_features': 500, 'vect__min_df': 10, 'vect__ngram_range': (1, 2)}	0,968	0,971	0,932	0,958	0,888	0,943	0,030
284	{'cls__C': 0.7, 'cls__loss': 'hinge', 'cls__max_iter': 500, 'vect__max_df': 0.5, 'vect__max_features': 500, 'vect__min_df': 10, 'vect__ngram_range': (1, 2)}	0,967	0,969	0,929	0,954	0,896	0,943	0,027
285	{'cls__C': 0.7, 'cls__loss': 'squared_hinge', 'cls__max_iter': 500, 'vect__max_df': 1.9, 'vect__max_features': 500, 'vect__min_df': 10, 'vect__ngram_range': (1, 2)}	0,967	0,969	0,929	0,957	0,884	0,941	0,032
285	{'cls__C': 0.7, 'cls__loss': 'squared_hinge', 'cls__max_iter': 500, 'vect__max_df': 0.5, 'vect__max_features': 500, 'vect__min_df': 10, 'vect__ngram_range': (1, 2)}	0,967	0,969	0,929	0,957	0,884	0,941	0,032

Como se muestra en la tabla anterior cada experimento posee un valor rank que hace referencia a que tan bien se desempeño en el entrenamiento, sin embargo hasta acá no se ha mostrado la mejor combinación de hiper-parámetros para el modelo SVM [18] planteado. La búsqueda en cuadrícula se aplica para calcular los mejores parámetros que se utilizarán para el modelo y es importante tener en cuenta que la búsqueda en cuadrícula puede ser extremadamente costosa desde el punto de vista computacional tomando tiempos apreciables en ejecución máquina. Finalmente los

mejores parámetros para el modelo se muestran en la siguiente tabla.

Hiper parámetro	valor identificado Con GridSearch
C	0.2
Loss	Hinge
Max_iter	500
Max_df	1.9
Max_features	1000
Min_df	10
Ngram_range	(1,1)

Tabla 3.2: Hiper parámetros modelo de máquinas de soporte vectorial

Con los parámetros del modelo entrenado ahora es posible clasificar cualquier texto de los que se han capturado en paralelo al entrenamiento y perfilar así el proceso de extracción de contexto geográfico, el desempeño de los mejores parámetros se puede ver en la tabla 3.3.

Tabla 3.3: Desempeño mejor modelo de parámetros

Rank	**Parámetros**	**split0**	**split1**	**split2**	**split3**	**split4**	**Media Test**	**Desviación Test**
1	{'cls__C': 0.2, 'cls__loss': 'hinge', 'cls__max_iter': 500, 'vect__max_df': 1.9, 'vect__max_features': 1000, 'vect__min_df': 10, 'vect__ngram_range': (1, 1)}	0,977	0,974	0,959	0,966	0,917	0,959	0,021

3.5.2. Matriz de confusión y validación

La matriz de confusión evalúa el rendimiento del modelo una vez entrenado, en esta se pueden identificar fácilmente los verdaderos negativos y verdaderos positivos.

predicción de Clases			
Clase Actual		0	1
	0	900	520
	1	582	530

Tabla 3.4: Matriz de confusión

La matriz muestra la clasificación, la primera fila diferenciada con el número cero son todos los tweets que no contienen información sobre incidencias viales, en este caso 900 se clasificaron correctamente estos comúnmente se denominan verdaderos

negativos, mientras que 520 fueron erróneamente clasificados y los llamaremos falsos positivos, la segunda fila considera textos que contienen incidencias viales, el primer valor refiere aquellos que se clasificaron erróneamente (falsos negativos) mientras 530 tweets se clasificaron correctamente, estos se conocen como verdaderos positivos.

3.5.3. Transferencia de aprendizaje y automatización

Como se dijo se capturaron en total 69.274 tweets con el conjunto de palabras semilla, los cuales podían o no contener información relevante para la investigación. Ahora que el modelo se encuentra entrenado y sabemos cómo se comporta la clasificación se puede ejecutar para cada uno de los registros. Del total el 45.68 % de los datos fueron clasificados como incidencias de tráfico para la ciudad de Bogotá lo que significa que tenemos 31.647 tweets que reportan incidencias y fueron clasificados como tal, esta cantidad es bastante grande pues oficialmente un año normal en la ciudad de Bogotá la autoridad de tránsito reporta un total de incidencias vehiculares de 34.990 con la dificultad que solo hay registro de accidentes de tránsito y son accesibles solo en forma consolidada a final de año o incluso en años posteriores, en cambio la información capturada por el algoritmo incluye información relevante en tiempo real, por ejemplo localización de un vehículo detenido en la vía, un semáforo sin funcionamiento o incluso una marcha, lo que hace que minutos o segundos después de ocurrida la incidencia sea posible extraerla de la red social y ubicarla en el espacio con sus coordenadas, localizar en el espacio no es una tarea para nada sencilla y como se dijo se trata de información no estructurada, por lo que se intentará extraer los elementos necesarios para geolocalizarla en el espacio en los siguientes apartados.

3.6. Extracción de contexto geográfico

Trabajar a escala urbana resulta demasiado interesante pues los trabajos documentados en el capítulo 2, marco de referencia, se enfocan en escalas muy generales y se alejan de la nomenclatura de ciudad, por ejemplo algunos de ellos recolectaban un lugar geográfico de una noticia, en ese caso solo se podía extraer la ciudad donde ocurría el hecho pero no se podía saber mucho más, pues en algunos casos las noticias solo informaban superficialmente, otros trataban lugares importantes de un país o ciudad, un parque o una plaza, por ejemplo la plaza de Bolívar, el inconveniente de estas abstracciones de información es que por ejemplo en Colombia cada ciudad puede tener más de un sitio geográfico con el nombre “simón bolívar” lo que lleva a ambigüedades ubicando un número indeterminado de lugares encontrados por un georreferenciador, esto es un problema y se sigue estudiando sobre este te-

ma de desambiguación, algunas iniciativas es geonames y what3words que con ideas disruptivas intentan dar solución.

Para este trabajo dada la escala urbana, se usaron expresiones regulares en base a las direcciones de la ciudad de Bogotá, donde se reporta por un lado la vía principal y luego la vía secundaria, así:

#	Vía Principal	Vía Secundaria
1	Av. Caracas	Calle 19
2	Av. Circunvalar	Calle 16
3	Carrera 60	Av. Esperanza

Tabla 3.5: Objetivos de recuperación

Pasando la estructura clave (Vía principal vs vía secundaria) a un georreferenciador para Bogotá, el 42.19 % de los elementos clasificados fue georreferenciado esto equivale 13.352 registros, además dicho georreferenciador hace posible obtener localidades y UPZ en la que ocurre el evento dando valor al de coordenadas (latitud y longitud). Lo anterior quiere decir que de un texto como:

"@BogotaTransito @Fontibon_Bogota @SectorMovilidad La Calle 13 o Avenida Centenario está totalmente trancada sobre la Carrera 123, por favor envíen unidades de tránsito urgente."

Se deben extraer los valores que permiten ubicar en el espacio esta incidencia, por un lado, la avenida calle 13 y la carrera 123, efectivamente el algoritmo extrae los valores necesarios. Por un lado especifica que la UPZ es Zona franca en la localidad Fontibón, lo cual es correcto y el par de coordenadas que devuelve son latitud 4.67904593 y longitud -74.17070604, la forma de hacer esto es identificar la estructura de la nomenclatura para Bogotá, de tal forma que el algoritmo se enfoque en buscar los valores que pueden ser direcciones y las pasa al georreferenciador, en este caso si el georreferenciador responde con "fallo la georreferenciación" quiere decir que el par de claves identificadas no corresponden a direcciones, depuran los resultados de clasificación.

Capítulo 4

PATRONES Y CORRELACIÓN ESPACIAL

Para dar solución a los objetivos 3 y 4 de la investigación, se procede a georeferenciar en un sistema de visualización geográfico. Como se ha dicho se parte de lenguaje plano (texto) y con el proceso hecho en el capítulo anterior ahora podemos hablar en términos geográficos de las dinámicas del fenómeno analizado, gracias al NLP y a la extracción de contexto geográfico.

En la imagen 4.1 se pueden visualizar todos los puntos en el espacio en un tamaño de punto pequeño para no saturar la imagen de información poco digerible, para el análisis existen varias formas de evidenciar patrones en los datos espaciales, esto dado que los puntos por si solos no dicen mucho. A simple vista se ve una distribución desordenada y sí se identifican conglomerados, pero solo se puede asegurar que existen de forma subjetiva pues no podemos calcular ningún valor de conglomerados en puntos distribuidos, al parecer no existen patrones pero si concentraciones, para saber si esto es cierto e identificar algún tipo de relación en la datos se va a identificar las correlaciones espaciales globales del conjunto de datos y luego las correlaciones locales que como resultado nos permite identificar 4 categorías muy importantes, pero esto se analizará más adelante.

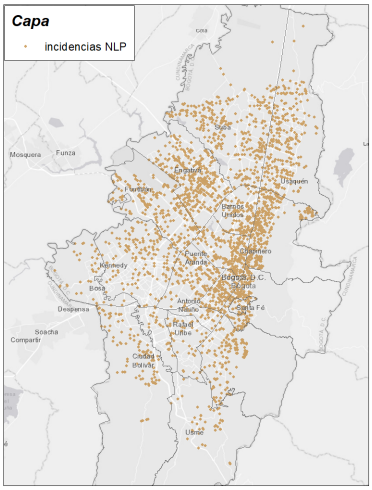


Figura 4.1: Incidencias Viales

El primer paso para avanzar en la identificación de patrones es dividir el espacio, la forma de hacerlo es crear una cuadricula de forma artificial a partir de las latitudes y longitudes, máximas y mínimas del conjunto de datos total, en este caso:

$$Latitud\_min,\ Longitude\_min,\ Latitud\_max,\ Longitud\_max = [-74.21577776,\ 4.47306594,\ -74.01140495,\ 4.81224183]$$

Con esto podemos decidir cuanto debe medir cada una de las áreas de la malla, para este trabajo y puesto que el análisis se hace a nivel urbano, se generan áreas cuadradas de 0.005°de lado, que es aproximadamente 555 metros de lado, como resultado el área de estudio tiene un total de 2.788 cuadrados iguales como se ve en la imagen 4.2.

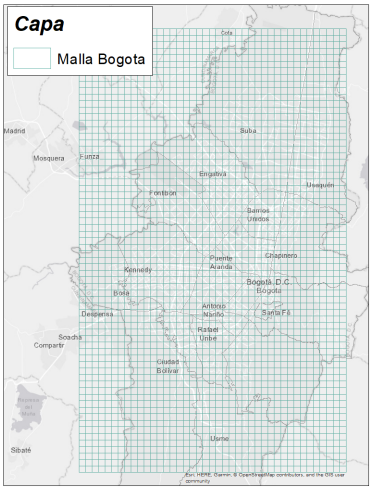


Figura 4.2: FishNet ó Grilla

El siguiente paso, es aprovechar la programación para ubicar cada uno de los puntos dentro de su correspondiente partición y contar cuantos eventos sucedieron en cada superficie de forma discreta, el objetivo de esto es dar el atributo de conteo a la malla en base a algún identificador de los polígonos, agrupar esta información es muy interesante pues ahora generalizamos un poco más el fenómeno y ahora podemos identificar en que partes puede existir un patrón.

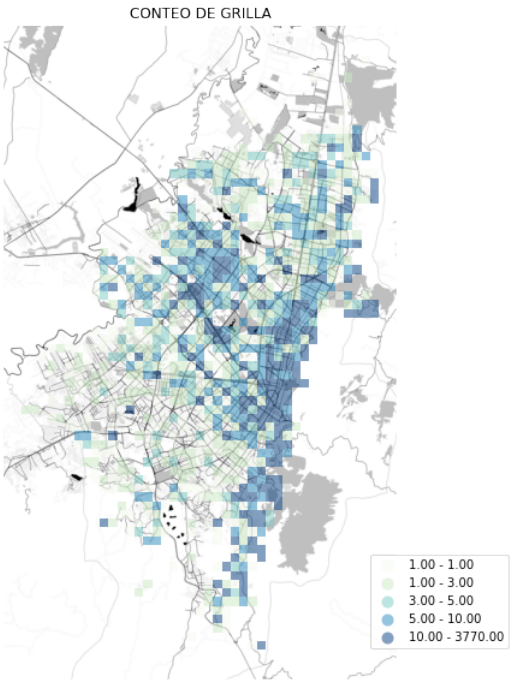


Figura 4.3: Conteos Sobre grilla

En la figura 4.3 se ve ahora de forma un poco más claro en donde se agrupa la información geográfica, la imagen anterior se encuentra clasificada en cuantiles por lo que cada partición agrupa una proporción bien definida e igual de los datos, esta es una primera aproximación muy válida, sin embargo ahora queremos saber más de los datos, en especial sí existen correlaciones espaciales entre estos, es decir, que tanto influyen los datos cercanos unos a otros, por esto se decide agrupar los incidentes en forma de cuadriculas contiguas.

4.1. Auto correlación espacial global

Una inspección visual del mapa de conteos nos permite buscar alguna estructura espacial, si la distribución de los conteos fuese aleatoria entonces nosotros no veríamos nada similar a clúster, sin embargo, se puede evidenciar clústeres altos, color oscuro, hacia la parte sur oriental con algunas partes centrales de la ciudad, también se

encuentran buenos clúster hacia las vías principales como av. Las Américas, calle 13 y calle 80, entre otras. Su contraparte de clústeres con valores bajos se encuentra hacia la parte sur occidente y noroccidente donde hay pocos patrones y con valores muy bajos. En este caso el análisis ha sido con ayuda de la lógica y nuestro cerebro, sin embargo, también puede caer en errores e identificar falsos positivos o patrones donde estadísticamente no existen.

El concepto de auto correlación relaciona aquellas dos cosas que no hace nuestro cerebro al tiempo en estos casos evitando la subjetividad, esta relaciona la similitud espacial con la similitud de un atributo, aunque hay muchas medidas de auto correlación espacial gran parte combina estos dos tipos, en resumen.

Para aplicar este método de auto correlación espacial global, lo primero es asignar unos pesos espaciales a nuestra grilla de datos, para este trabajo y dado que nos importan todas las vecindades en cada superficie se ha tomado el criterio de la reina, la cual es una referencia a la reina en ajedrez cuyos movimientos no tienen restricción de dirección, en este caso se ha usado la librería de Python científico y espacial Pysal, la cual posee mucho algoritmos usados en análisis espaciales complejos.

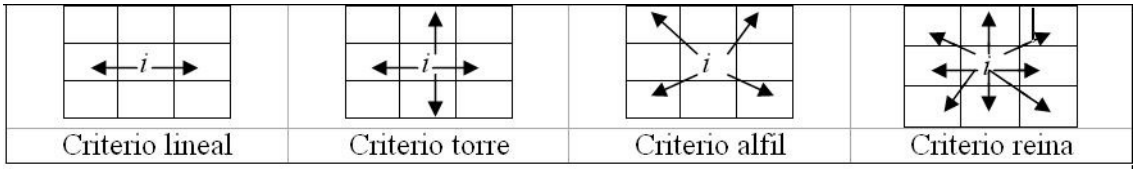


Figura 4.4: Criterios de vecindad

Lo siguiente una vez calculados la matriz de pesos espaciales que identifica si dos vecinos, i y j son geográficamente similares, pero también se necesita una medida de similitud de atributos para relacionarlo con el concepto de similitud espacial, la variable que realiza dicha relación se suele llamar "spatial lag" o retraso espacial que es una variable derivada, entonces para i el retraso espacial se define como:

$$y_i = \sum_j w_{i,j} y_j \quad \text{Retraso Espacial} \tag{4.1}$$

Calculando este retraso lo podemos poner en una imagen y analizar nuestra variable derivada y que servirá a la hora de encontrar patrones.

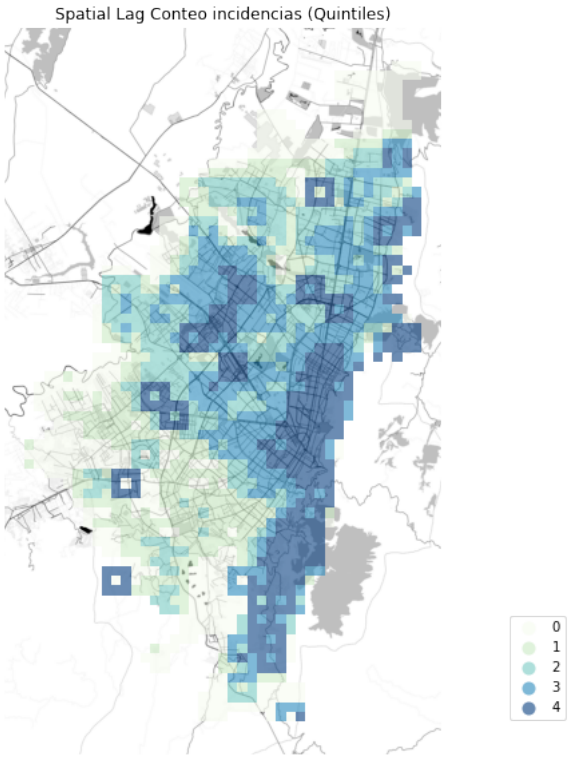


Figura 4.5: Spatial lag de conteos de incidencias

Lo que podemos ver en la figura 4.5 es que se conservan parte de los cluster mencionados anteriormente, pero ahora hay clusteres oscuros más grandes, veamos dicha comparación en quintiles con la clasificación inicial.

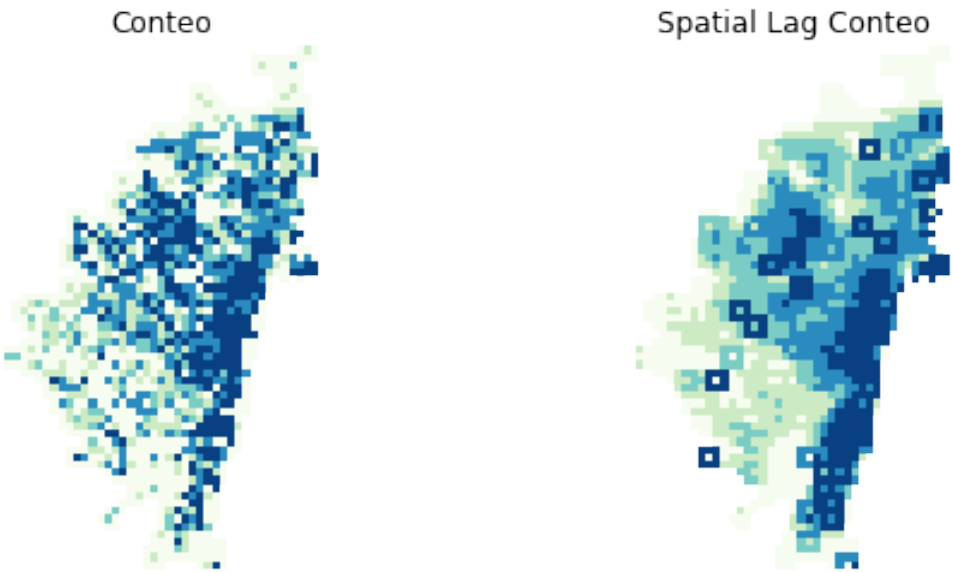


Figura 4.6: comparación conteos y retraso espacial en quintiles

El siguiente paso es asignar correlaciones altas y bajas comparando la variable conteo, es decir la convertimos en una variable binaria, en nuestro caso tenemos 713 celdas que están por encima de la media, entonces la distribución espacial se puede mostrar en forma binaria como sigue.

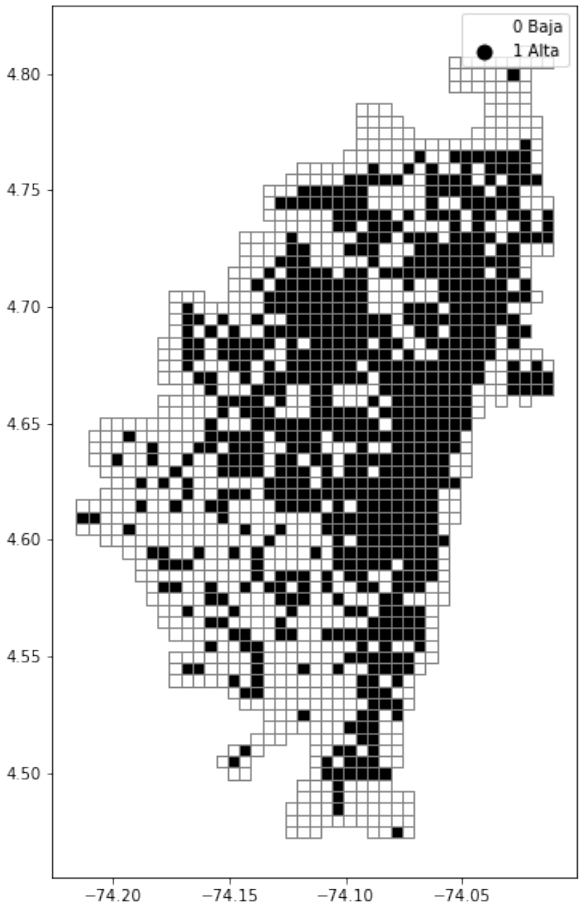


Figura 4.7: Resultado de correlación global

La anterior es nuestra correlación espacial global, como se ve es muy general y solo definimos correlaciones altas y bajas, pero este mapa nos dice que casi toda la ciudad en el caso de incidencias de tránsito tiene correlación fuerte con el espacio en el que ocurre, esto como un primer análisis exploratorio funciona, pero esta investigación pretende ir más allá por lo que una medida más específica, espacialmente hablando, es la auto correlación espacial local.

4.2. Auto correlación espacial local

Para identificar patrones nos ayudaremos de técnicas geoestadísticas, una de estas es ampliamente usada y se conoce como estadística local de Moran y su gráfico de dispersión, para mostrar su utilidad fijemos la atención en la siguiente imagen:

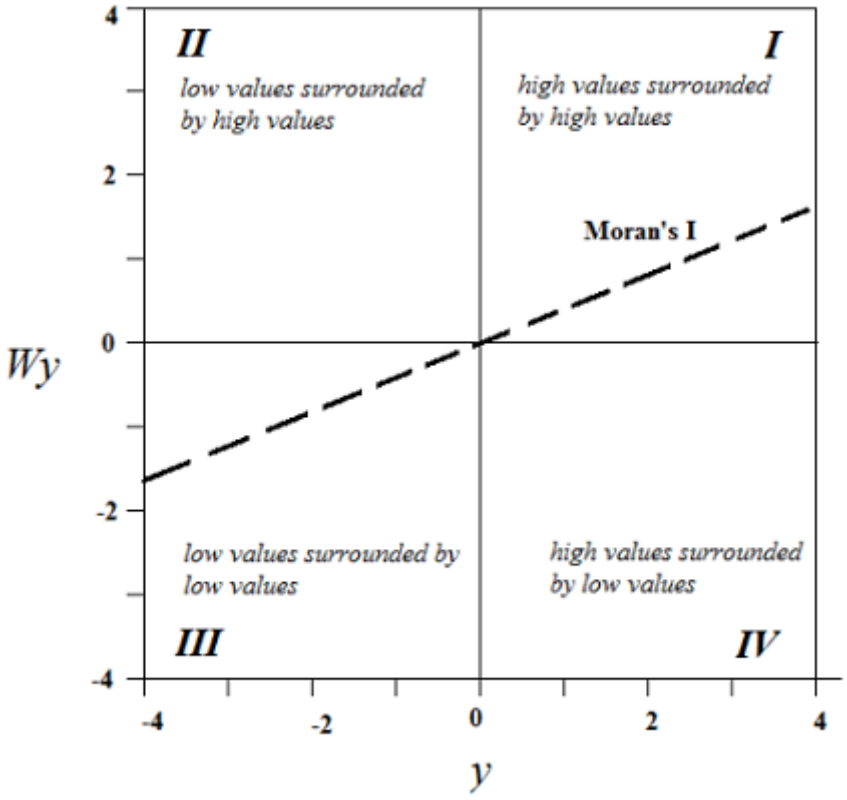


Figura 4.8: Lectura del diagrama de dispersión de Moran

Anselin, nos explica el funcionamiento de los gráficos de dispersión de Moran y su división por cuadrantes. Consideremos un fenómeno cualquiera, el primer cuadrante nos muestra valores espaciales altos del fenómeno que son rodeados por valores altos es decir hay fuerte presencia del fenómeno, el cuadrante 2 son valores bajos pero que se encuentran rodeados por valores altos, formando algo como una dona si lo viéramos de forma espacial. El tercer nivel agrupa valores bajos y el cuarto cuadrante tiene valores altos rodeados de valores bajos. Para el caso de la investigación el cálculo de el diagrama de dispersión es el siguiente:

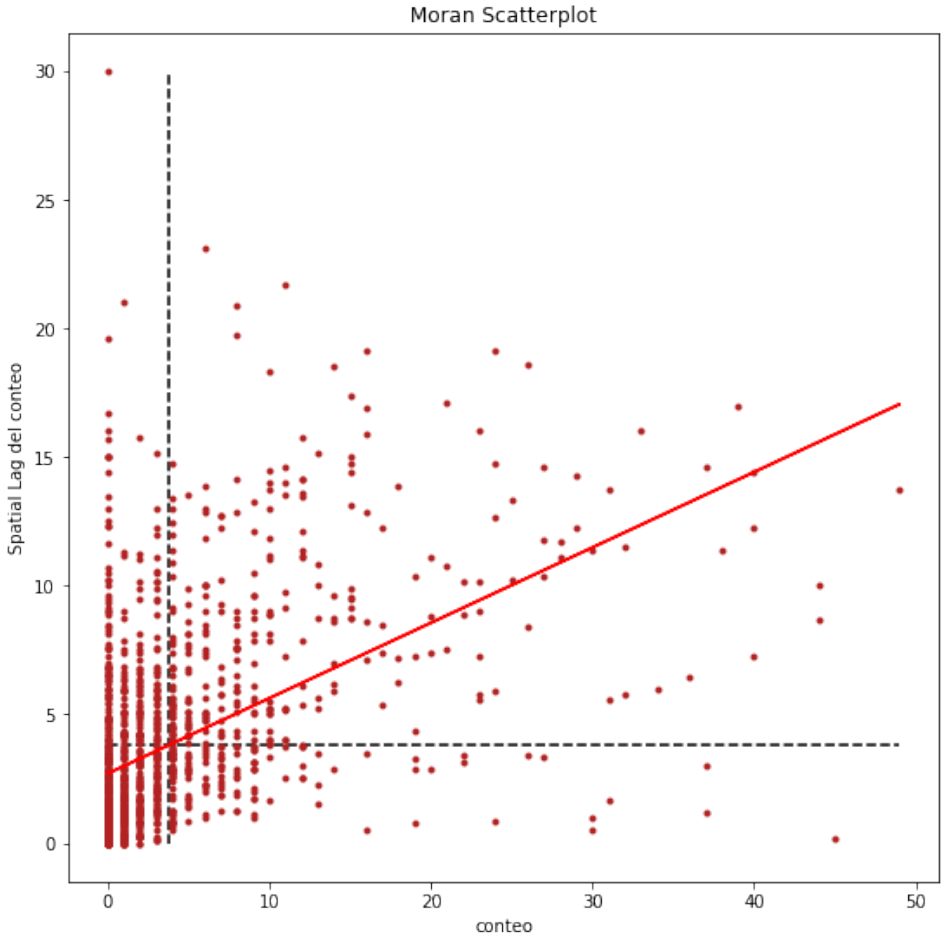
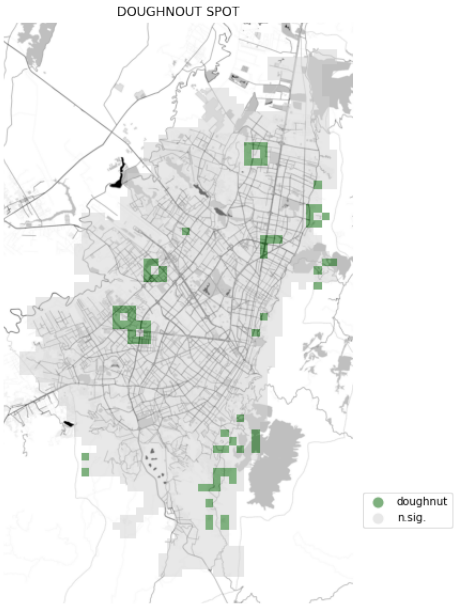


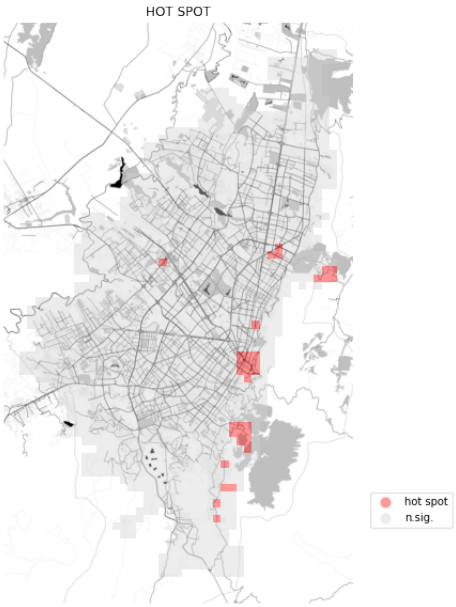
Figura 4.9: Diagrama de dispersión de Moran

En este se puede ver la distribución de los datos dentro de los cuadrantes, existiendo datos en todos, según nuestro mapa de dispersión gran parte de los datos estan en valores bajos, pero si existen clústeres de valores altos que son los más importantes a la hora de tomar decisiones sobre las actividades viales de la ciudad, ahora sería interesante verlos en un mapa.

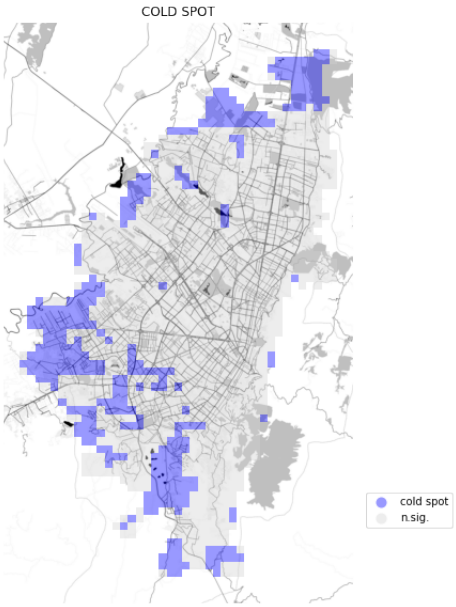
Las imágenes siguientes tienen el mismo orden que el mapa de dispersión de moran, por lo que ahora si sabemos espacialmente donde se localizan los puntos calientes, puntos fríos, tipo dona y de diamante, en los que podemos tomar decisiones, esta técnica es comparativamente mejor que los mapas de calor que en algunos casos son mal usados, por ejemplo en este trabajo hacer un mapa de calor no seria para nada interesante, pues lo que hace un mapa de densidad como ese es contar los valores cercanos y agruparlos y en aquellos sitios en donde no exista presencia del fenómeno suele interpolar los datos con otros próximos, esto desemboca en malas interpretaciones por lo que no es aconsejable usarlos con fenómenos que no son continuos.



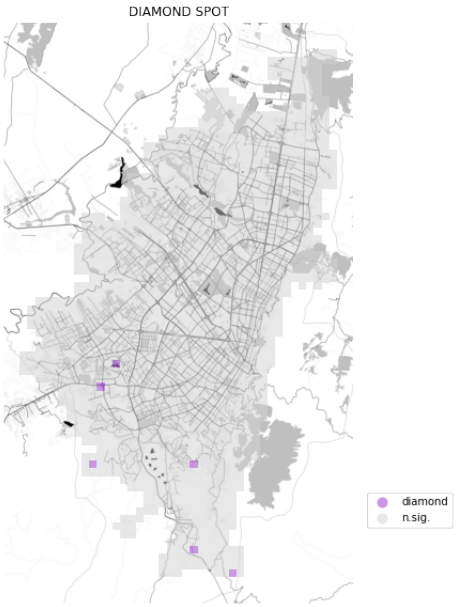
(a) Puntos dona



(b) Puntos calientes



(c) Puntos fríos



(d) puntos diamante

Figura 4.10: Clasificación de clusters según auto correlación local

En resumen, se han identificado exactamente cuales son los clústeres de la ciudad en términos de recolección de información a partir de NLP y repetición en zona, se crea una malla que delimita la zona en donde fue posible obtener el contexto geográfico y se calcularon las auto correlaciones espaciales globales y locales.

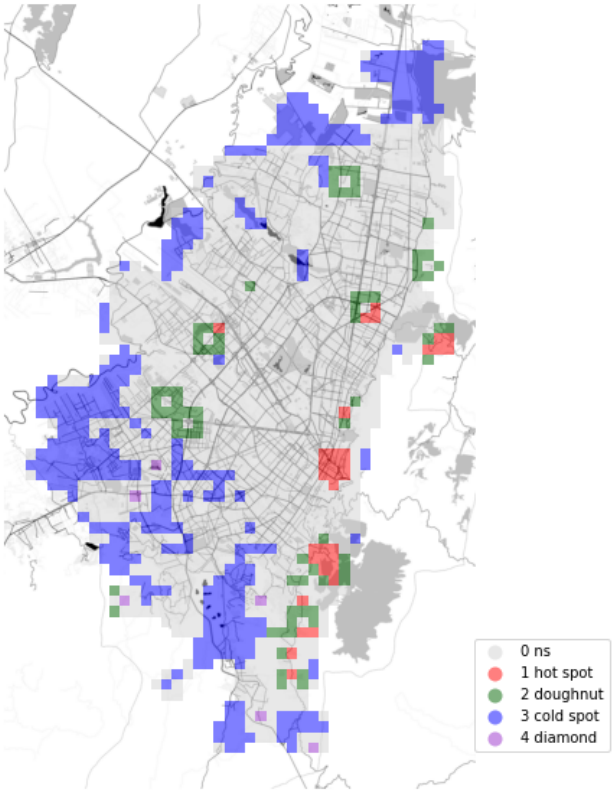


Figura 4.11: Resultado auto correlación espacial local

Inicialmente se evidenciaron patrones en los datos con una clasificación por cuantiles, pero nuestra capacidad de identificar patrones mediante el campo visual nos llevaba a identificar falsos positivos en los datos espaciales por lo que una técnica de la geoestadística ahora nos dice exactamente donde los valores son significativos en términos estadísticos.

4.3. Validación de patrones con 2019

Gracias a la política de datos abiertos adoptada en Colombia es posible adquirir los datos del año inmediatamente anterior junto a un consolidado, en este caso para hacer un acercamiento a la realidad y a una base de datos oficial. Se hizo exactamente el mismo proceso para identificación de un patrón espacial pero únicamente para el año 2019 con datos ya georreferenciados y depurados por las entidades pertinentes, recuerde que los datos extraídos con NLP cuentan con datos de 2019 y los patrones deberían compartir algunas características, y es lo que vamos a corroborar con ayuda de la auto correlación y diagrama de dispersión de moran en la siguiente figura.

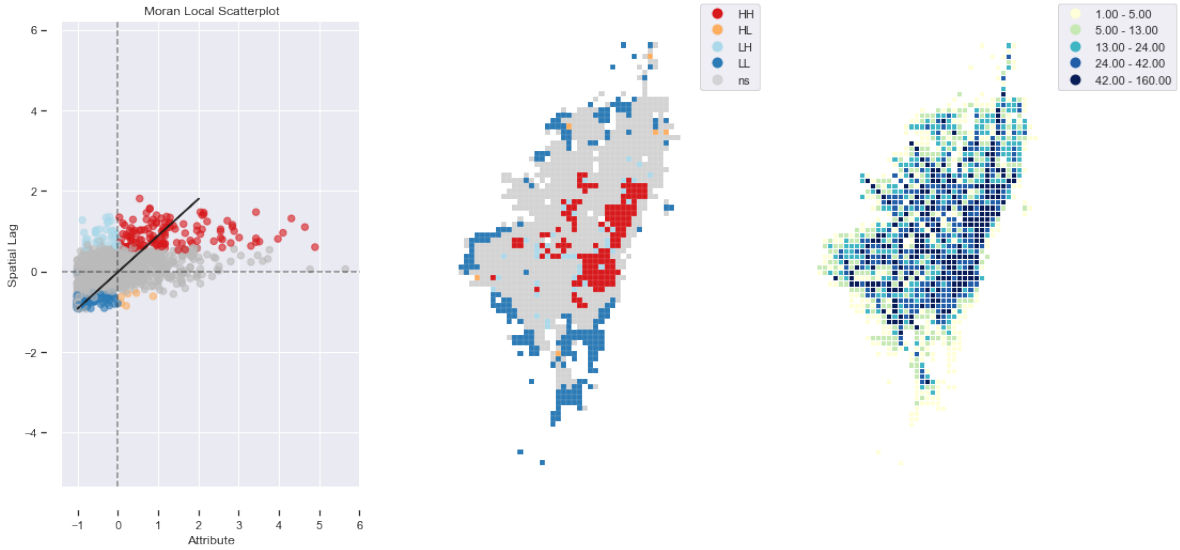


Figura 4.12: Resultado auto correlación local 2019 - datos libres

El conjunto de datos 2019 descargado de la fuente oficial y datos abiertos, cuenta con un total de 29.115 registros y en este se puede ver los focos de incidentes viales, puntos calientes, puntos fríos, tipo dona y diamante, notando que los agrupamientos son más grandes que los concluidos con NLP sin embargo se evidencias similitudes y características equiparables. En la imagen siguiente se puede ver que se conserva la estructura con focos calientes hacia el oriente y focos fríos hacia las afueras de la ciudad.

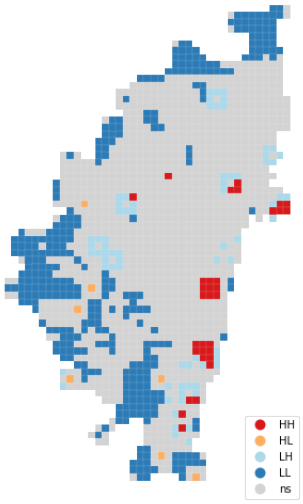


Figura 4.13: Auto correlación local 2020 - datos NLP

Es de resaltar que la imagen fue generada con datos capturados en 2019 y 2020, pero dado que el 2020 es un año particular puede que influyera directamente en la identificación de estos patrones perdiendo algo de densidad en cada celda.

Capítulo 5

ANÁLISIS DE RESULTADOS

La etapa de recolección de datos cubrió la totalidad de 11 meses sumando un total de 69.274 textos con más del triple de metadatos que describen de donde proviene esta información, desde el usuario, la fecha de creación del texto entre otras cosas. Los datos agrupados pasan a la etapa de aplicación de algoritmos de inteligencia artificial y NLP para hacer una clasificación, en este caso incidencias viales, la salida de la clasificación fueron 31.647 textos identificados como reportes de incidencia vial, es decir el 45.68 % del conjunto inicial. Como se aclara en el capítulo 3 está información describe posibles incidencias, para verificar que sean textos que contienen algún tipo de referencia espacial y pasan al proceso de extracción de foco geográfico. Recuperado este foco es esencial pasarlo por un georreferenciador para darles un lugar en el espacio, la proporción de rescate de información espacial en base a la entrada de esta etapa fue de 42,19

La tabla 5.1 cuenta la concentración de los eventos totales a la vez qué muestra valores altos hacia la localidad de santa fe, suba y san Cristóbal, de manera inicial se podría pensar en patrones hacia esas localidades pues los eventos superan la media.

Localidad	Incidencias
Santa Fe	4181
Suba	1291
San Cristóbal	1244
Chapinero	1030
Usaquén	1005
Fontibón	973
Kennedy	870
Engativá	743
Usme	453
Teusaquillo	336
Ciudad Bolívar	232
Puente Aranda	209
Barrios Unidos	202
Los Mártires	192
Bosa	123
Rafael Uribe Uribe	115
Candelaria	94
Tunjuelito	28
Antonio Nariño	26

Tabla 5.1: Extracción por localidades

El análisis de auto correlación global y local dan cuenta sobre la influencia y presión espacial de las incidencias viales sobre algunas zonas de la ciudad y ejes viales principales, esto fue posible evidenciarlo dado el potencial del procesamiento natural del lenguaje para extracción de eventos que tienen lugar en un espacio y afectan a la comunidad en general y que además sirven a un interés común que es el monitoreo de una ciudad que posibilita una mejor administración y toma de decisiones a partir de datos. Esta sensibilidad del modelo a espacializar datos no estructurados a un espacio geográfico se encaminan a una ciudad sostenible e inteligente. La imagen muestra los puntos significativos en cuanto a patrones, por ejemplo tenemos puntos calientes hacia el oriente de la ciudad en vías importantes como la caracas con calle 26 y sus alrededores, o ejes principales como la Boyacá, autopista norte y calle 13, vitales para la entrada de suministros y aunque no se aborda en este trabajo las consecuencias de aglomeración en estas zonas concurridas disminuye la calidad de vida de los habitantes y alarga los tiempos de transporte entre dos puntos de la ciudad. De igual forma se pueden ver relaciones en cuanto a focos en documentos como [41] y [40] e incluso se demuestra la influencia de estos puntos sobre la vía con trabajos como [46].

Al comparar un año normal como el 2019 ratifica los patrones y valida la extracción en su configuración espacial, es posible que la coyuntura histórica de pandemia

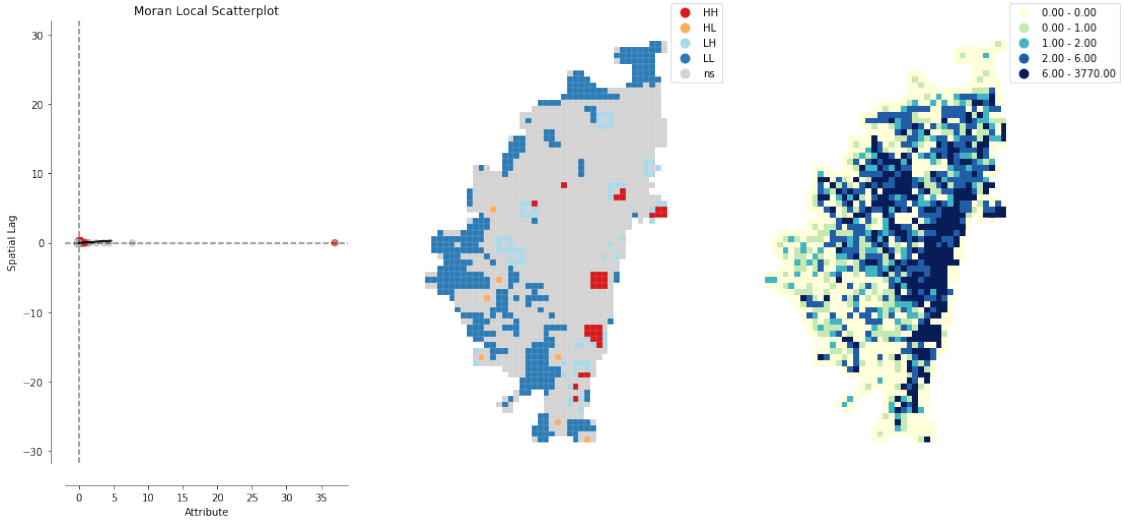


Figura 5.1: Resumen auto correlación espacial local

por el COVID-19 afectara los resultados pues la ciudad no se encuentra a su máxima capacidad, por lo que manifestaciones, mantenimiento de infraestructura o accidentes se han visto disminuidos pues ruedan menos vehículos en las vías arteriales. Es claro que hay algunas vías que escapan a la extracción de foco geográfico y esto se debe especialmente a la forma como los usuarios escriben en una red social, pese a esto, se ha logrado abordar con resultados satisfactorios la ambigüedad que puede tener alguien al escribir una localización urbana, además el clasificador se desempeño de forma apropiada pues como han concluido otros autores SVM se comporta mucho mejor que clasificadores como KNN o Naive bayes [3] en inteligencia artificial.

En resumen, la adquisición de información por vías no convencionales apoyadas en inteligencia artificial reduce costos y dependencia a otras tecnologías, el modelamiento de esta solución de seguro necesita apoyarse en nuevas herramientas o modelos de AI, pero la metodología acá planteada muestra resultados prometedores que aterrizan muy bien en el análisis de eventos espaciales. A pesar de que 2020 fue un año atípico y la captura disminuyó debido a las continuas cuarentenas los patrones espaciales persisten a los identificados en el 2019, por lo menos aquellos que poseen mayor ocurrencia de incidencias viales, validando en cierta medida los resultados y patrones identificados en este trabajo.

Capítulo 6

CONCLUSIONES

Como se mostró en la sección anterior de análisis de resultados, la recuperación hecha con las técnicas de minería de texto muestra un porcentaje de recuperación especifico en el problema de 46 % para información capturada con un conjunto inicial de palabras escogidas según su frecuencia en pequeños textos de interés al problema, bajo esta consideración de palabras iniciales como semilla en la captura de datos los textos resultantes podían contener una o varias palabras pertenecientes al grupo de control de captura, sin embargo un poco más de la mitad de los textos no fueron clasificados como reporte de incidencia vial lo que puede indicar que el conjunto inicial de palabras de captura debería ampliarse en su diversidad identificando aquellas palabras que puedan escapar al grupo inicial y aporten algunos datos más de captura. Descrito lo anterior el proceso de recuperación de entidades geográficas dentro de la ciudad mostró un 43 % de geo codificación exitosa, este resultado se debe interpretar de dos formas, la primera es que a pesar de que se toman todos los textos clasificados como incidencia no todos se reportan con alguna localización o referencia, se suma de forma paralela la posibilidad que la extracción del foco no pueda georeferenciar una dirección por un posible error de quien reporta o que algún reporte use referencias espaciales poco conocidas o no específicas, por ejemplo “frente al éxito de la calle 80” en este caso es poco especifico pues hay más de un almacén sobre dicha vía por lo que la referenciación de ese evento no es posible, pese a lo anterior lograr un 43 % de información georreferenciada hace apreciable la cantidad de datos y al final si evidencian patrones espaciales similares a los de años anteriores. También es válido aclarar que el año 2020 del cual se poseen datos es especialmente atípico por ser el primer año de pandemia pues se presume que la cantidad de datos fueron menos a cualquier otro año, es de esperar que posteriormente a la pandemia o con una mejor gestión de está los datos capturados sean más y los patrones espaciales se asemejen aun más.

Las herramientas geo estadísticas evaluadas como la auto correlación espacial global y local fueron esenciales a la hora de visualizar patrones espaciales que depende fuertemente de su entorno pues al contrario de otros métodos como los mapas de calor en formato raster, esté sí considera los pesos espaciales que acompañado de los mapas de dispersión de moran deja ver esos puntos del espacio que se encuentran en niveles altos del fenómeno y que demás el entorno inmediato muestra índices altos de auto correlación, lo mismo sucede con puntos fríos y aquellos que no son relevantes al análisis son triviales a la hora de evaluar fenómenos espaciales.

Desde el año 2019 se han presentado avances prometedores en el área de procesamiento natural del lenguaje, sin embargo los modelos presentados son cada vez más abrumadores en tamaño para investigadores independientes, incluso para instituciones de educación superior, cada modelo presentado evidencia el uso intensivo de poder de cómputo y en algunos casos hacen necesario el uso de HPC (High perfomance computing) que paraleliza tareas de código entregando a cada core de hardware una tarea especifica agilizando el entrenamiento de modelos de inteligencia artificial, pese a que esto haría más rápido el entrenamiento de modelos de AI, los recursos de hardware no están disponibles para todos los investigadores y en la mayoría de los casos necesitan que quien ejecuta el modelo sea capaz de paralelizar código, en algunos casos python paralelizando con ayuda de Dask o con lenguajes como C, C++ o fortran a través de una API como Openmp que permite compartir memoria. Ahora bien, la figura 6.1 presenta la evolución y tamaño de los modelos de NLP que en su mayoría son financiados por empresas privadas, el eje Y representa el número de parámetros del modelo partiendo de modelos razonablemente manejables a otros como T-NLG que en 2020 entreno un modelo con 17 billones de parámetros y solo manejable en granjas de servidores con grandes cantidades de dinero, quien batió ese record y no esta documentado en la imagen por la forma repentina y reciente de su aparición fue GPT-3 de la empresa OpenAI que cuenta con 175 billones de parámetros y cuyas demostraciones con modelos reducidos muestran avances muy prometedores en la generación de texto, además OpenAI se ha reservado de no liberar su modelo completo.

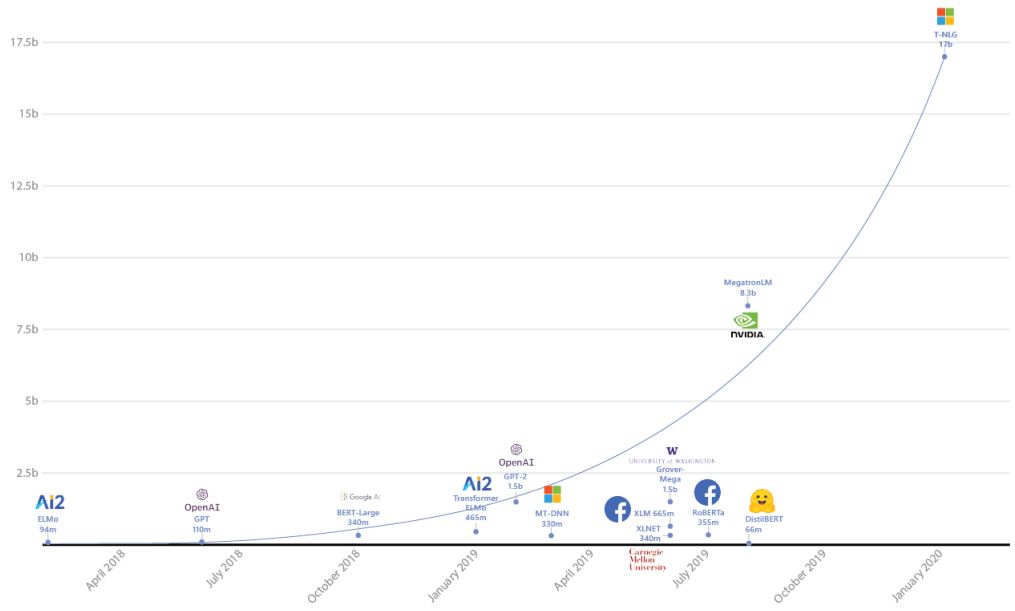


Figura 6.1: parámetros modelos 2018 - 2020, fuente: Microsoft

Como se dijo las capacidades de cómputo y procesamiento paralelo permiten generar modelos como el de GPT-2 o GPT-3, modelos de generación de texto basado en probabilidad, haciendo sencillo escribir un mail o tener conversaciones artificiales con un chatbot y de paso mejores experiencias a un usuario final, este tipo de modelos son un ejemplo de inyección de capital y de computación de alto desempeño, como investigador en este tema creo fuertemente que el procesamiento del lenguaje nos seguirá impresionando con más características que pueden ayudar a personas de diferente espectro social y cultural, es el momento ideal para incentivar la extracción de foco geográfico con los nuevos avances y llegar a aplicaciones variadas de impacto y alto crecimiento.

De forma adicional se mostró que la extracción de foco geográfico se puede llegar a relacionar con la abstracción que hacen las ciudades sobre las localizaciones, es decir nomenclatura vial, además esto es posible extenderlo a vías nacionales o municipales haciendo uso del código de la vía y el kilómetro, esta podría ser una buena extensión, sin embargo, el campo del NLP se encuentra en auge y los avances parecen ser mejores cada mes, la salida del algoritmo GPT-3 muestra un generador de texto que funciona a base de probabilidades lo que quiere decir que de un conjunto de palabras escoge las mejores palabras (mayor probabilidad de secuencia), además conserva el contexto en una conversación simulada. Un ejercicio adicional hecho en este trabajo fue el uso del modelo reducido de GPT-3 con una pequeña exploración de la respuesta del algoritmo, según estas pruebas GPT-3 logra discernir entre conjuntos de palabras que hacen referencia a lugares y responde según dicho contexto

sin decirle exactamente que es un "lugar", esto se puede explicar porque este algoritmo fue entrenado con casi toda la información de internet, mayormente en idioma ingles. En conclusión, al no ser objeto de este trabajo clarificarlo y al estar fuera del alcance se pude decir que GPT-3 puede funcionar en la correcta identificación de un lugar geográfico, ahora bien estos algoritmos de NLP no entregan coordenadas pero se puede pensar en algo complementario a los diccionarios geográficos teniendo palabras simultaneas y lejanas de un mismo texto como referencia, con lo que agilizaría la búsqueda y escalabilidad de estas soluciones. En este caso hay prometedores avances en la extracción de contexto geográfico que pueden luego decantarse en chatbots geográficos o descripción de zonas geográficas automatizadas, las posibilidades y aplicaciones pueden ser sumamente innovadoras y dependerá de los nuevos investigadores.

A la fecha se han presentado avances en la identificación del foco geográfico, algunos trabajos han intentado extraer el foco geográfico a través de diccionarios geográficos, llamados también Gazetteers, sin embargo, el problema de ambigüedad continua pues muchos lugares geográficos pueden tener el mismo nombre, incluso en la misma ciudad, otras soluciones intentan generar un ranking geográfico asignando un valor que diferencie entre mayormente buscados o de mayor importancia según búsquedas, lo cual es similar a un ataque de fuerza bruta en donde el resultado depende exactamente de la forma en que esta escrito el objeto buscado, a pequeña escala esto puede funcionar relativamente bien pero pensando en un problema global para cualquier ubicación en el mundo la solución deja de ser escalable pues cada texto debería ser indexado y aquellas palabras susceptibles a ser localizadas con coordenadas son buscadas en una gran base de datos y sí se habla de miles de textos el esfuerzo aumenta de forma exponencial, ante esto es posible ayudarse de ideas como What3words que pretenden que una localización sea definida con apenas 3 palabras, lo que puede ser explorado de forma interesante. Para futuras investigaciones sería interesante implementar las tecnologías que emergieron los últimos años dado su desempeño excepcional en la generación y reconocimiento de texto, además, la parte espacial se puede enriquecer en la georreferenciación y desambiguación de localizaciones, un flujo de trabajo como el mostrado en esta investigación pone sobre la mesa la importancia de empalmar la inteligencia artificial y la geoestadística como análisis espacial a la masiva generación de información diaria que en algunos casos se ve subutilizada en este punto donde la combinación de técnicas puede generar crecimientos acelerados en términos económicos y sociales, mejorando la calidad de vida y potenciando las capacidades del país hacia el exterior.

BIBLIOGRAFIA

Agencia Nacional de seguridad Vial. (2012). Metodología para la identificación de sectores críticos de accidentalidad en zonas urbanas, *2*, 25.

An, S., Yang, H., Wang, J., Cui, N. & Cui, J. (2016). Mining urban recurrent congestion evolution patterns from GPS-equipped vehicle mobility data. *Information Sciences*, *373*, 515-526. https://doi.org/10.1016/j.ins.2016.06.033

Anastácio, I., Martins, B. & Calado, P. (2009). Classifying documents according to locational relevance. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5816 LNAI*, 598-609. https://doi.org/10.1007/978-3-642-04686-5_49

Applegate, D. L., Bixby, R. E., V, C. & Cook, W. J. (2006). The Traveling Salesman Problem: A Computational Study. http://search.ebscohost.com/login.aspx?direct=true%7B%5C&%7Ddb=nlebk%7B%5C&%7DAN=390512%7B%5C&%7Dsite=eds-live

Caicedo, E. & Lopez, J. (2017). *Una aproximacion práctica a las redes neuronales artificiales*. Universidad del Valle.

Clark, K., Luong, M.-T., Le, Q. V. & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, arXiv 2003.10555, 1-18. http://arxiv.org/abs/2003.10555

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. & Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context, arXiv 1901.02860, 2978-2988. https://doi.org/10.18653/v1/p19-1285

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, (1950), arXiv 1810.04805. http://arxiv.org/abs/1810.04805

Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA, O'Reilly Media.

Green, B. F., Wolf, A. K., Chomsky, C. & Laughery, K. (1961). Baseball: An automatic question-answerer. *Proceedings of the Western Joint Computer Con-*

ference: Extending Man's Intellect, IRE-AIEE-ACM 1961, 219-224. https://doi.org/10.1145/1460690.1460714

Gu, Y., Qian, Z. & Chen, F. (2016). From Twitter to detector: Real-time traffic incident detection using social media data. *Transportation Research Part C: Emerging Technologies*, *67*, 321-342. https://doi.org/10.1016/j.trc.2016.02.011

H. J. Payne & S. C. Tignor. (1978). Freeway incident detection algorithms based on decision tree with states. *Transportation Research Record*, 30-37. http://onlinepubs.trb.org/Onlinepubs/trr/1978/682/682-005.pdf

Jones, K. S. (1994). Natural Language Processing: A Historical Review. *Current Issues in Computational Linguistics: In Honour of Don Walker*, *3*, 3-16. https://doi.org/10.1007/978-0-585-35958-8_1

Joos, M., Locke, W. N. & Booth, A. D. (1956). Machine Translation of Languages: Fourteen Essays. *Language*, *32*(2), 293. https://doi.org/10.2307/411008

Karimzadeh, M., Pezanowski, S., MacEachren, A. M. & Wallgrün, J. O. (2019). GeoTxt: A scalable geoparsing system for unstructured text geolocation. *Transactions in GIS*, *23*(1), 118-136. https://doi.org/10.1111/tgis.12510

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C. & Socher, R. (2019). CTRL: A Conditional Transformer Language Model for Controllable Generation, arXiv 1909.05858, 1-18. http://arxiv.org/abs/1909.05858

Lample, G. & Conneau, A. (2019). Cross-lingual Language Model Pretraining, arXiv 1901.07291. http://arxiv.org/abs/1901.07291

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, arXiv 1909.11942, 1-17. http://arxiv.org/abs/1909.11942

Lane, H., Howard, C. & Max Hapke, H. (2019). *Natural Language Processing In action*.

Leidner, J. L. (2008). Toponym resolution in text: Annotation, evaluation and applications of spatial grounding of place names. *Institute for Communicating and Collaborative Systems School of Informatics University of Edinburgh*, *41*(2), 292. https://doi.org/10.1145/1328964.1328989

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach, (1), arXiv 1907.11692. http://arxiv.org/abs/1907.11692

Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de la Clergerie, É. V., Seddah, D. & Sagot, B. (2019). CamemBERT: a Tasty French Language Model, *2*arXiv 1911.03894. http://arxiv.org/abs/1911.03894

Martins, B. & Silva, M. J. (2005). A Graph-Ranking Algorithm for Geo-Referencing Documents.

Minsky, M. & Papert, S. (1969). Perceptrons: expanded edition. *MIT Press Cambridge MA*, *522*, 20. https://doi.org/10.1016/S0019-9958(70)90409-2

Peregrino, F. S., Tomás, D. & Llopis, F. (2013). Una aproximación basada en corpus para la detección del foco geográfico en el texto. *Procesamiento de Lenguaje Natural*, *50*, 69-76.

Perkins, J., Chopra, D. & Hardeniya, N. (2016). Natural Language Processing : Python and NLTK. http://www.nltk.org/book

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018). Improving Language Understanding by Generative Pre-Training. *OpenAI*, 1-10. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language%7B%5C_%7Dunderstanding%7B%5C_%7Dpaper.pdf

Purves, R. S., Clough, P., Jones, C. B., Hall, M. H. & Murdock, V. (2018). Geographic information retrieval: Progress and challenges in spatial search of text. *Foundations and Trends in Information Retrieval*, *12*(2-3), 164-318. https://doi.org/10.1561/1500000034

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. (2018). Language Models are Unsupervised Multitask Learners.

Sanh, V., Debut, L., Chaumond, J. & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, arXiv 1910.01108, 2-6. http://arxiv.org/abs/1910.01108

Stephanedes, Y. J. & Chassiakos, A. P. (1993). Freeway incident detection through filtering. *Transportation Research Part C*, *1*(3), 219-233. https://doi.org/10.1016/0968-090X(93)90024-A

Tensorflow. (2019). Word embeddings — TensorFlow Core. Recuperado el 13 de junio de 2020, desde https://www.tensorflow.org/tutorials/text/word%7B%5C_%7Dembeddings

Thancanamootoo, S. (1988). AUTOMATIC DETECTION OF TRAFFIC INCIDENTS ON A SIGNAL-CONTROLLED ROAD NETWORK.

Twitter Inc. (2020). API reference index — Twitter Developers. Recuperado el 22 de junio de 2020, desde https://developer.twitter.com/en/docs/api-reference-index

Vargas, W., Mozo, E. & Herrera, E. (2012). Análisis De Los Puntos Más Críticos De Accidentes De Transito En Bogotá. *Revista de topografia Azimut*, *4*(4), 7-22. http://revistas.udistrital.edu.co/ojs/index.php/azimut/article/view/5741/7212

Werbos, P. J. (1994). The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. & Brew, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing, arXiv 1910.03771. http://arxiv.org/abs/1910.03771

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding, (NeurIPS), arXiv 1906.08237, 1-18. http://arxiv.org/abs/1906.08237

Yu, R. & Abdel-Aty, M. (2013). Utilizing support vector machine in real-time crash risk evaluation. *Accident Analysis and Prevention*, *51*, 252-259. https://doi.org/10.1016/j.aap.2012.11.027

Zhang, Z., He, Q., Gao, J. & Ni, M. (2018). A deep learning approach for detecting traffic accidents from social media data. *Transportation Research Part C: Emerging Technologies*, *86*(November 2017), 580-596. https://doi.org/10.1016/j.trc.2017.11.027