Análisis y visualización de peticiones, quejas y reclamos ciudadanos*

Rocío Hubert

Universidad Nacional del Sur, Departamento de Ciencias e Ingeniería de la Computación, San Andrés 800, B8000FTN Bahía Blanca, Argentina

Resumen Citymis Community es una aplicación de gobierno electrónico que permite a los municipios escuchar y responder a los reclamos y pedidos de los ciudadanos. Como consecuencia del exitoso uso de Citymis Community en múltiples municipios, se generan a diario grandes volúmenes de datos que requieren ser analizados por los agentes públicos que correspondan. El análisis inteligente de las peticiones, quejas y reclamos ciudadanos puede mejorar el nivel de la coordinación entre servicios y ayudar a las autoridades pertinentes en el proceso de toma de decisiones. El uso de nuevas técnicas de visualización y de minería de datos permite mejorar la presentación y posterior análisis de esta información. En este trabajo presentamos CitymisVis, una herramienta que extiende Citymis Community aplicando métodos especialmente desarrollados para realizar clustering sobre datos geolocalizados y explorar reportes de manera visual e interactiva. CitymisVis permite realizar un análisis visual sobre áreas geográficas afectadas en mayor medida por diferentes tipos de problemas y ofrece también datos estadísticos de la información recolectada por medio de representaciones navegables. De este modo, CitymisVis ayuda a entender qué problemas requieren de mayor atención, y permite reflejar de manera adecuada las proporciones de las diferentes problemáticas.

1 Introducción y motivaciones

Las peticiones, quejas y reclamos (PQR) ciudadanos proveen una valiosa fuente de información sobre los problemas relacionados con la infraestructura y con los servicios de los municipios. La aplicación de tecnologías de información y comunicación en la administración pública para recolectar y procesar PQR ciudadanos provee canales apropiados para la comunicación y coordinación electrónica entre todas las partes interesadas (ciudadanos, funcionarios del gobierno, etc.) [1], [2], [3]. En el contexto particular de gestión inteligente de ciudades, se han desarrollado numerosas plataformas para escuchar y responder PQR de ciudadanos [4], [5], [6]; sin embargo, estas plataformas solo proveen una solución parcial al problema de analizar eficiente y eficazmente largos volúmenes de datos que son generados diariamente por los usuarios. En algunos casos, los PQR de los ciudadanos corresponden a dificultades aisladas, mientras en otros casos están relacionados con problemas más grandes; algunos datos podrían

* Parte de esta tesis de grado fue presentada y publicada en "10th International Conference on Theory and Practice of Electronic Government"(ICEGOV 2017), llevada a cabo en New Delhi, India, 5-7 Marzo de 2017.

revelar problemas persistentes que impactan áreas geográficas específicas o un tipo de servicio en particular, que deben ser identificadas y resueltas por las autoridades pertinentes priorizando las demandas más urgentes. Utilizando los avances en tecnologías de visualización y minería de datos se pueden desarrollar herramientas que faciliten este análisis de PQR ubicados geográficamente.

Citymis Community es una aplicación EGOV mediante la cual los municipios pueden escuchar y responder en tiempo real en una red colaborativa, reduciendo significativamente los tiempos de respuesta y mejorando la calidad de interacción con los ciudadanos. En este trabajo presentamos CitymisVis, una herramienta que provee diferentes funcionalidades interactivas para el análisis visual y la exploración de PQR denunciadas mediante la aplicación Citymis Community. De este modo, CitymisVis ayuda a reconocer un rango de problemas en la infraestructura comunitaria y en la prestación de servicios, facilitando una visión de las proporciones de los problemas identificados; resultando en una valiosa ayuda en el proceso de toma de decisiones de las autoridades pertinentes. En este trabajo ilustraremos las diferentes funcionalidades de CitymisVis usando reportes generados por usuarios de San Isidro, una municipalidad de aproximadamente 350,000 habitantes ubicada cerca de la ciudad de Buenos Aires.

2 Citymis

Citymis Community (https://citymis.co/ - Citizen Centric EGOV) es un ecosistema de aplicaciones web y móviles experto en la gestión holística de servicios municipales, diseñado tanto para ciudadanos como para agentes públicos municipales y desarrollado por Mismatica Management (http://www.mismatica.management), que permite a los usuarios reportar problemas desde sus computadoras o celulares. Esta aplicación constituye un medio por el cual las municipalidades pueden escuchar y responder problemáticas ciudadanas en tiempo real, y los ciudadanos pueden reportarlas con la seguridad de que llegarán a manos de funcionarios públicos. Esta sección presenta las principales características de Citymis Community, basado en el material presentado en [7].

Figura 1: Modelo Citymis.

Para describir el flujo de mensajes y/o tareas entre el ciudadano y el funcionario público hasta la resolución del problema, se comenzará desde el punto en el que se realiza el reclamo de una problemática.

Al momento de realizar un reporte, el usuario puede distinguirlo especificando una categoría del problema, como puede ser “Alumbrado Público”, “Agua Corriente”, “Calles y otros” o “Higiene Urbana”, entre otras que incluye la aplicación para cubrir los servicios que provee una municipalidad. A su vez, cada una de estas categorías tiene una serie de subcategorías para una mejor especificación de la problemática observada. Por ejemplo, dentro de “Alumbrado Público”se incluyen categorías relacionados a luminarias apagadas e intermitentes, pedido de una nueva luminaria o pedido de reparación, entre otras.

El ciudadano también deberá ingresar la calle y la altura donde se encuentra el problema o, alternativamente, especificar la ubicación por medio de un mapa. La existencia del mapa no sólo provee la facilidad de especificar la ubicación geográfica exacta, sino también de encontrar y examinar otros problemas reportados en la misma área. Además, el usuario puede escribir un comentario y proporcionar una foto para una mejor descripción de la problemática. La aplicación también requiere de información de contacto del usuario, a fin de poder notificarlo sobre cualquier posible cambio en el estado del reclamo.

Desde una perspectiva operacional, la municipalidad se divide en subregiones geográficas, llamadas “áreas de asignación”. Esta división permite asignar los reportes a empleados específicos (inspector, equipo de mantenimiento, etc.) para resolver o verificar los problemas reportados. Cada reclamo es asociado en primera instancia a una “unidad de gestión”, que es básicamente una ubicación geográfica (latitud, longitud, calle y altura) y se le asigna el estado de “pendiente de moderación”. En este punto, el reporte puede ser aceptado, en cuyo caso se le cambia el estado a “recibido”, o puede ser descartado, recibiendo el estado de “rechazado”. Una vez que el reclamo es aceptado, el siguiente paso es resolver el problema a fin de obtener el estado de “cumplido” y terminar el proceso. Cada vez que el estado de un reclamo cambia, el usuario que realizó el reporte recibe una notificación por email y es invitado a realizar nuevos comentarios para brindar información adicional que pueda ser de ayuda para resolver el problema, o simplemente para dar su opinión respecto al servicio.

Figura 2: Modelo lineal de PQR tradicionalmente utilizado.

A diferencia del modelo tradicional de PQR utilizado (que puede verse en la Figura 2), el modelo de Citymis permite reportar y atender reclamos en tiempo real, llevando a un 70 % de reducción en los pasos operativos. El modelo de Citymis puede verse en la Figura 1.

3 CitymisVis

Con el propósito de obtener información útil sobre la información recolectada mediante la aplicación de Citymis Community, se desarrolló CitymisVis, una herramienta web con diversas funcionalidades que permiten al ciudadano y a los empleados municipales analizar mapas y estadísticas de manera interactiva. Para el desarrollo de CitymisVis se utilizaron las herramientas presentadas en el Apéndice B. Para ilustrar el uso de esta herramienta, se proveen ejemplos creados en base a los reportes de la municipalidad de San Isidro. El conjunto de datos utilizado por la herramienta contiene 68,368 reportes realizados por ciudadanos que usan la aplicación móvil, la aplicación web de Citymis Community o un número de teléfono gratuito para reportar el problema. CitymisVis tiene acceso a varios atributos asociados con cada reporte, como el tipo de servicio, el tipo de problema y la ubicación del reporte (latitud, longitud, calle y altura). Además, tiene acceso a los comentarios de los usuarios (que es la descripción del reporte).

En la página de inicio, CitymisVis muestra una serie de imágenes que rotan cada cierto intervalo de tiempo, que introduce y permite acceder a las distintas funcionalidades del sitio. Estas funcionalidades son los tres mecanismos principales que provee CitymisVis para explorar y analizar la información: mapas de calor, clusters y datos estadísticos, que son descritos a continuación.

3.1 Mapas de calor

Un mapa de calor es una representación gráfica de datos que utiliza colores para indicar el nivel de actividad; en el caso de CitymisVis, un mapa de calor permite diferenciar la cantidad de PQR en una región geográfica determinada. Por ejemplo, la Figura 3 presenta un mapa de calor para todos los servicios de San Isidro.

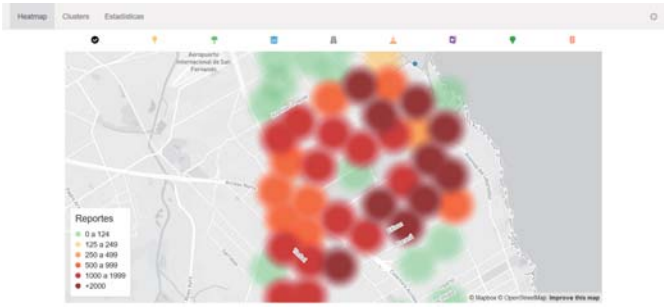

Figura 3: Mapa de calor que muestra las diferentes concentraciones de reportes.

Para la creación y representación visual del mapa de la ciudad, se utiliza una directiva provista por Mapbox, indicando las coordenadas geográficas centrales de la ciudad, el nivel de zoom inicial, y el nivel mínimo y máximo de zoom permitidos.

Para el funcionamiento y la correcta visualización del mapa de calor, se debe asignar un source al mapa y establecer la opción de cluster como true. El contenido de este

source será el GeoJSON que contenga la información geográfica requerida por Mapbox. La escala de valores para los grados de calor utilizada para el mapa de calor está definida en un archivo JSON (llamado layers), donde se define un arreglo que contiene pares detallando la cantidad de puntos requeridos para formar un grado y el color asignado a ese grado. Este archivo es utilizado por la función heatmap provista por Mapbox. En el mapa, se incluye una sección de referencia para los diferentes gradientes, utilizándose colores más cálidos a medida que aumenta la concentración de reportes en una misma zona del mapa. De este modo, los colores del mapa varían en un rango desde verde (ligeramente afectado), pasando por amarillo (moderadamente afectado), hasta llegar a rojo (muy afectado).

Utilizando directivas provistas por Mapbox, se agregan nuevas capas al mapa, asociadas al source asignado y en base al arreglo de layers definido con anterioridad. De este modo, quedan vinculados los clusters generados por Mapbox con las layers, y al agregarse las capas al mapa, se puede apreciar el mapa de calor según la escala utilizada.

La herramienta desarrollada ofrece opciones de zoom in y zoom out, haciendo posible visualizar la ciudad en su totalidad o enfocarse en áreas específicas. Además, al llegar al máximo nivel de zoom in podrán verse los reportes individuales como puntos de color azul. Adicionalmente, es posible visualizar un mapa de calor para el conjunto total de servicios, o seleccionar y ver el mapa de calor para un servicio específico. Para esto, al cargar la página, se recuperan los GeoJSON generados para cada servicio (los diferentes sources), de modo que al cambiar de opción solo reste borrar las capas asociadas al source utilizado hasta el momento, y agregar al mapa las capas que corresponden a la opción elegida. La Figura 4 presenta un mapa de calor para el servicio de “Alumbrado Público”de San Isidro.

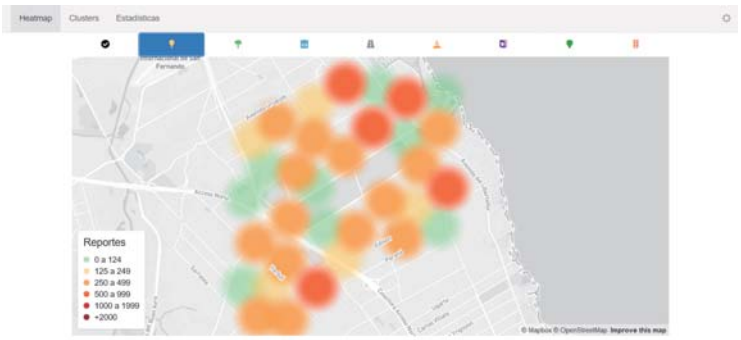

Figura 4: Mapa de calor que muestra las diferentes concentraciones de reportes asociados al servicio de “Alumbrado Público”.

Para la generación de estos sources, en el servidor existe una rutina encargada de obtener de la base de datos la información asociada a los reclamos y generar los archivos GeoJSON que correspondan (uno por cada servicio y uno aparte para el total de los reclamos), siguiendo las reglas especificadas para este formato. El nombre utilizado para estos archivos está sujeto al id del servicio, de modo que sea fácil y directo acceder

a ellos una vez que el usuario elige una opción para visualizar. Esta rutina del servidor no se ejecuta cada vez que se abre la página o se elige una opción de visualización diferente. Sería conveniente que se ejecute cada cierto periodo de tiempo y en background, a fin de actualizar los datos de los reportes que se reflejarán en el mapa de calor. Para la elección del periodo de tiempo entre actualizaciones se debería considerar la fluctuación de los reportes a lo largo del tiempo y valorar cuán actualizados se desean los datos.

Los mapas de calor permiten distinguir diferentes concentraciones de reportes a lo largo de diferentes zonas geográficas, ofreciendo una ayuda valiosa para reconocer las áreas más afectadas.

3.2 Clusters

Esta funcionalidad permite visualizar agrupaciones de PQR de diferentes dimensiones sobre un mapa de la ciudad. Estas agrupaciones o clusters son generados por el algoritmo DBSCAN (ver Apéndice B).

Para la creación y visualización del mapa, se usa la misma directiva de Mapbox que usan los mapas de calor. Al igual que las representaciones de los mapas de calor, los mapas de agrupaciones permiten visualizar clusters generados sobre el conjunto total de servicios, u observar un servicio en específico. Además, existe la opción de mostrar aquellos puntos que han quedado fuera de las agrupaciones formadas (outliers), visualizados como markers individuales sobre el mapa. Esto puede verse en la Figura 5.

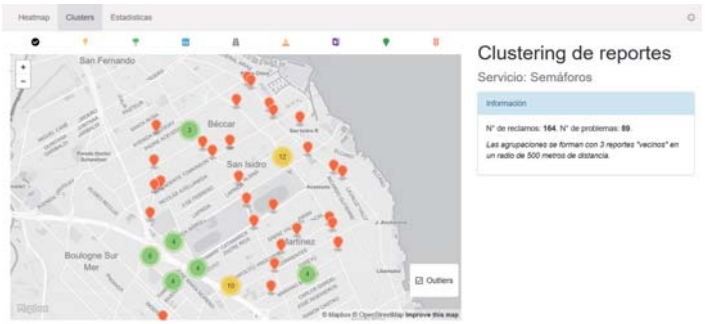

Figura 5: Mapa de la ciudad que muestra las diferentes agrupaciones para el servicio “Semáforos”, y los outliers como markers de color rojo.

La herramienta también permite visualizar clusters de reportes a diferentes niveles de granularidad. Estas granularidades definen distintos valores para los parámetros asociados al algoritmo DBSCAN (épsilon y minPts), que dependerán de la cantidad de reclamos con los que se quieran computar clusters. Los valores están definidos en un archivo JSON en donde se determinará una correspondencia entre cantidad de reclamos y parámetros de DBSCAN asociados.

De este modo, cuando el usuario selecciona un servicio y una determinada granularidad, se recupera del servidor —de una carpeta asociada al servicio elegido y a los parámetros épsilon y minPts vinculados a esa granularidad— los archivos GeoJSON generados previamente.

Cada uno de esos archivos representa agrupaciones diferentes encontradas por el algoritmo DBSCAN de ELKI, con los parámetros determinados y los datos relacionados a los reclamos del servicio escogido.

El plugin Markercluster provisto por Mapbox es el utilizado para representar los clusters calculados sobre el mapa de la ciudad. Para utilizarlo, se crea un objeto MarkerClusterGroup por cada cluster encontrado por el algoritmo DBSCAN. A las agrupaciones se le asigna un color para indicar si contiene una gran (rojo), mediana (amarillo) o pequeña (verde) cantidad de reportes. La Figura 6 muestra diferentes agrupaciones obtenidas a partir de conjunto de datos analizados, con los diferentes colores representando diferentes concentraciones de reportes.

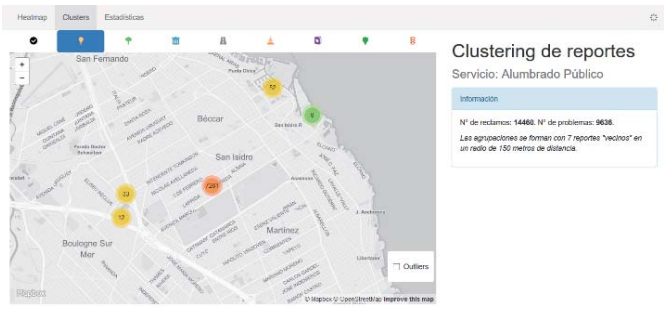

Figura 6: Clustering de reportes asociado al servicio de “Alumbrado público”.

Estos objetos MarkerClusterGroup, que representan a los clusters, tienen diferentes características de interés. Una de ellas es que permiten visualizar subclusters de una agrupación al hacer clic sobre ella, haciendo uso del algoritmo hierarchical greedy clustering. Esto puede seguir realizándose a diferentes niveles de zoom hasta llegar al máximo permitido, donde se podrán observar los reportes que los conforman en sus ubicaciones geográficas específicas, representados mediante markers.

De esta forma, el algoritmo DBSCAN se usa para identificar los clusters principales (agrupaciones de mayor precisión) y luego se combina con el algoritmo de hierarchical greedy clustering ofrecido con Mapbox, para identificar subclusters de manera dinámica y eficiente.

Otra característica de MarkerClusterGroup es que permite apreciar visualmente los límites de un cluster particular al pasarle el mouse por encima, “dibujandoün polígono creado a partir de las coordenadas geográficas de los reportes que conforman los vértices del polígono (los límites del cluster). Para la implementación de la página web, se obtuvo ese polígono generado para que perdure más allá del mouse over. Al seleccionar un cluster, se dibuja un polígono que cubre todos los reportes en ese cluster. De

este modo, al seguir realizando la misma acción sobre subclusters, se puede observar un conjunto de polígonos anidados que están asociados a un cluster y sus subclusters. Además, al realizar zoom out, se reagrupan los subclusters en la agrupación que componen y deja de visualizar el polígono correspondiente a esa agrupación.

Conjuntamente a la visualización del mapa, la página tiene una segunda sección destinada a ampliar la información. En principio, indica qué servicio y granularidad fueron elegidos, además del número de problemas encontrados y de reclamos generados, donde un mismo problema puede tener varios reclamos asociados. Esta sección brinda información adicional al seleccionar un determinado cluster o marker.

Al hacer clic sobre una agrupación en el mapa, además de sus subclusters y polígonos, en la sección de la derecha, se generará un gráfico de torta —usando Simple pie chart— que mostrará las proporciones de subcategorías afectadas únicamente dentro de ese cluster, como puede verse en la Figura 7. Para generar este gráfico, se obtendrán y recorrerán los reportes de la agrupación para generar dinámicamente un archivo JSON con el formato especificado por Simple pie chart.

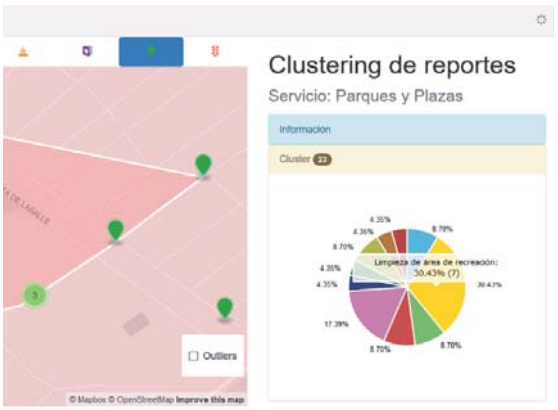

Figura 7: Gráfico de torta correspondiente a un cluster seleccionado en el servicio de “Parques y Plazas”.

En el lado izquierdo de la Figura 8 pueden observarse los polígonos resultantes de aplicar hierarchical greedy clustering sobre un cluster de reportes asociados al servicio de “Alumbrado Público”. En el lado derecho de la misma figura, hay información asociada al servicio y cluster seleccionado, acompañado por un gráfico de torta que muestra las proporciones de reportes que pertenecen a diferentes categorías del servicio analizado. Al pasar el mouse sobre las secciones individuales del gráfico se pueden identificar las subcategorías asociadas a ese cluster. En el ejemplo de la Figura 8 pueden apreciarse la incidencia de las distintas subcategorías en el cluster para el servicio de “Alumbrado Público”; en particular, el 62,50 % de los reportes dentro de ese cluster corresponden a “Luminaria apagada”.

Además de poder visualizar clusters y subclusters, y los límites geográficos que abarcan, se puede inspeccionar un reporte en particular haciendo clic sobre el marker

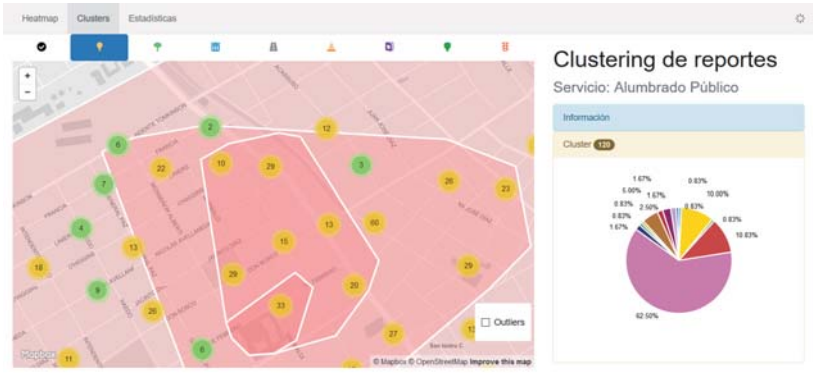

Figura 8: Hierarchical greedy clustering y polígonos anidados (izquierda) y gráfico de torta de las subcategorías (derecha) para el servicio de "Alumbrado Público".

que lo representa. Esta acción provoca que se abra un pop up que indica el servicio y la subcategoría a la que corresponde; y en la sección de la derecha de la página, podrá verse información como la ubicación y los comentarios dejados por los usuarios respecto a ese reclamo. Esta funcionalidad es ilustrada en la Figura 9.

Como ocurre con los archivos utilizados por los mapas de calor, los archivos asociados a las agrupaciones generadas por DBSCAN para todas las combinaciones de servicios y valores de parámetros de entrada al algoritmo, no se realizan ante cada pedido del usuario, sino que son generados de manera premeditada por el servidor. El algoritmo desarrollado por ELKI genera en una carpeta los archivos asociados a cada cluster encontrado por DBSCAN con los parámetros asignados, y, otra rutina en el servidor, los recorre para generar los archivos GeoJSON que son los que finalmente usa Mapbox. Para determinar cada cuánto se ejecutarán estas rutinas del servidor, deberá tenerse en cuenta tanto la fluctuación en la generación y resolución de reclamos como el valor de visualizar clusters más actualizados.

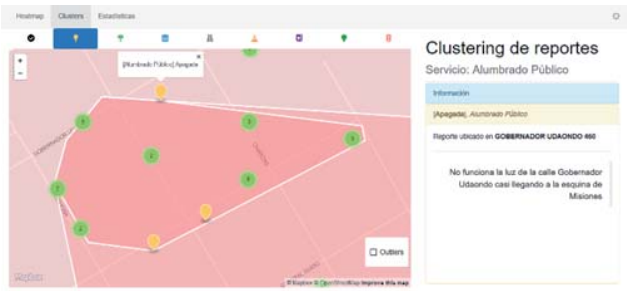

Figura 9: Inspección de un reporte específico mediante una funcionalidad que muestra, sobre la derecha, la información asociada al reporte seleccionado.

La funcionalidad provista por CitymisVis para poder observar agrupaciones de reportes y sus límites en el mapa, ofrece una valiosa ayuda para definir estrategias que mejoren los niveles de coordinación de servicios. En particular, permite diseñar una mejor logística para atender a las demandas de los ciudadanos mediante la consideración del tipo de servicio y su cercanía geográfica.

3.3 Datos estadísticos

CitymisVis también genera informes estadísticos utilizando todo el conjunto de datos recolectados por Citymis. Dichos informes son representados visualmente por medio de zoomable sunburst chart [8] de d3. Este gráfico permite presentar información organizada por jerarquía, donde el centro del gráfico corresponde al total de servicios (y de reportes provistos) y las secciones alrededor corresponden a los diferentes servicios y subcategorías. Para determinar el tamaño de cada sección, se le asigna un valor a cada subcategoría de cada servicio que se corresponde con la cantidad de reportes asociados a la subcategoría en cuestión. De este modo, los diferentes valores de cada subcategoría determinan el valor del servicio que conforman. A su vez, los valores calculados por los servicios conforman el valor global del total de los servicios (en el centro del gráfico).

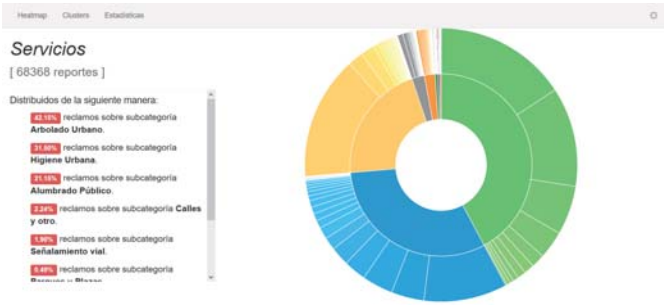

Figura 10: Sunburst chart para los reportes asociados a todos los servicios y sus subcategorías.

La Figura 10 presenta un sunburst mostrando las proporciones de reportes relacionados con todos los servicios y sus subcategorías en la municipalidad de San Isidro. Junto al gráfico hay una sección de información en donde pueden observarse los porcentajes de reportes generados sobre cada uno de los servicios. En el caso de un servicio en particular, se mostrarán sus subcategorías, su proporción dentro de ese servicio y su proporción respecto al total de reportes. Estas proporciones se calculan utilizando la cantidad de reportes realizados para cada subcategoría de cada servicio. Además, tanto la sección de información (abajo del título del servicio o subcategoría) como el gráfico (al mantener el mouse sobre una determinada sección) muestran la cantidad de reportes generados para el total de los servicios, un servicio o una subcategoría determinada.

Se pueden identificar las categorías y subcategorías en el gráfico pasando el mouse sobre las secciones. Una característica especial de esta implementación de sunburst

es que permite realizar zoom en cualquier arco o subsección que se cliquee, pudiendo disminuir el zoom al hacer clic en el centro del gráfico. La Figura 11 ilustra esta característica presentando información estadística y su representación gráfica correspondiente sobre la categoría de “Higiene Urbana”. En esta misma figura, se puede observar la característica que ofrece la herramienta para identificar las secciones del gráfico que se corresponden con los items de la lista en la sección de información. Al pasar el puntero por encima de algún item, la sección correspondiente se colorea de negro. Además, al hacer clic sobre el ítem, se realizará zoom in tanto en el gráfico sobre la sección elegida, así como también en la información asociada a esa misma porción. Para acompañar esta navegación por fuera del gráfico, se tienen dos botones que permiten volver a la sección anterior, así como también volver directamente a la totalidad de los servicios.

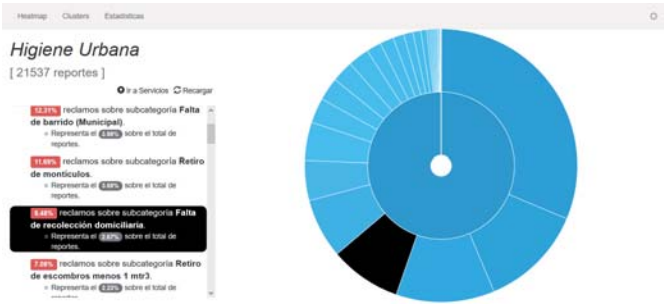

Figura 11: Información estadística para el servicio de Higiene Urbana, la sección elegida se resalta con negro y corresponde a la subcategoría de “Falta de recolección domiciliaria”.

Para que sea de utilidad a funcionarios públicos, es necesario que el archivo JSON que utilizan el gráfico y la sección de información sea generado de manera que esté lo más actualizado posible, visualizándose solo las problemáticas que no han sido resueltas y su incidencia en cada subcategoría, servicio o sobre el total.

El uso de estos gráficos resulta en una forma intuitiva de presentar información estadística de una manera visual y jerárquica que es fácil de navegar. Como consecuencia, ofrece una herramienta adecuada para tener una visión clara de las dimensiones de los problemas en una municipalidad basada en sus correspondientes categorías de servicios y subcategorías asociadas a esos servicios.

4 Trabajo relacionado

Muchas plataformas fueron desarrolladas para ofrecer comunicación entre gobiernos y ciudadanos. Ejemplos de estas plataformas pueden verse en [4], [5], [6], [9] y [10]. En [4] se analiza la evolución de los conceptos relacionados con la participación ciudadana, y el grado de e-participación a nivel local y nacional de Portugal, determinando en qué áreas se logró mayor participación y la distribución de esa participación entre hombres, mujeres, entidades y desconocidos, en base a la iniciativa de e-participación

promovida por el gobierno llamada “O meu movimento”. Por otro lado, en [5] se analizan 20 aplicaciones orientadas a resolver problemas urbanos complejos. Para cada una de ellas, se examinan los objetivos que persigue, la naturaleza de la información que utiliza, el motivo que llevó a su creación, y los desafíos urbanos que intenta resolver (como transporte y servicios públicos, salud, seguridad, entre otros). En [6] se propone un framework llamado E-Government 2.0, que integra los procesos, recursos, back offices y front offices de los sistemas en línea a fin de obtener un gobierno electrónico participativo orientado a las partes interesadas, utilizando como casos de estudio a Corea, Antigua y Barbuda, y Ecuador para validarlo. El análisis presentado en [9] plantea los objetivos de prestar servicios centrados en el ciudadano, que implica el diseño de los servicios desde el punto de vista del usuario en lugar del gobierno. Describe luego la experiencia de los autores al trabajar en proyectos de gobernanza electrónica en distintos estados de India (Goa, Delhi, Jharkhand, Karnataka y Andhra Pradesh). Finalmente, en [10] se describe CityEye, una plataforma que reúne operaciones, sensores y retroalimentación ciudadana mediante un panel de control web y una aplicación móvil basada en servicios, reduciendo la distancia entre ciudadanos y los proveedores de servicios, haciendo que los procesos de mantenimiento urbano sean más legibles a través de paneles de información en tiempo real e integrando bucles de retroalimentación activados por el usuario a través de sus dispositivos móviles.

Las plataformas desarrolladas permiten a los ciudadanos reportar problemas relacionados a la prestación de servicios como salud, educación, transporte público y seguridad, entre otros. Al igual que concluyeron algunos de los trabajos mencionados, el mayor beneficio que proveen esas plataformas es la reducción del tiempo de respuesta y las mejoras en la calidad de la prestación de servicios, como indican [11] y [12]. En [11] se analiza cómo pueden aplicarse tecnología avanzada de información, análisis y sistemas de pensamiento para desarrollar un enfoque de los servicios más centrado en el ciudadano, mostrando guías para mejorar diferentes aspectos de la ciudad, como pueden ser salud, transporte, higiene, seguridad y educación, y análisis de casos de estudio para cada uno de estos aspectos. Por otro lado, [12] ofrece una revisión de trabajos realizados por investigadores sobre la relación de gobernanza electrónica con varias dimensiones como la estrategia, la arquitectura empresarial y la interoperabilidad, indicando que las iniciativas de gobierno electrónico que aprovechan el potencial de las tecnologías de la información y la comunicación han tenido una gran influencia positiva en la habilitación y transformación de los servicios gubernamentales en muchos países.

Varios estudios han intentado obtener información respecto a los problemas y tendencias predominantes basado en la información obtenida de los reportes de ciudadanos, como puede observarse en [13] y [14]. En [13] se analizan reclamos de servicios realizados en la ciudad de Boston durante un año (2010-2011) y, usando análisis geoespacial y regresión binomial negativa, se investigan las posibles disparidades por raza, educación e ingresos en la solicitud de servicios. Por su lado, [14] analiza cómo las decisiones de diseño de los sistemas de retroalimentación ciudadana afectan la interacción entre ciudadanos y entre los ciudadanos y el gobierno, mediante el análisis de teoría espacial y fundamentada sobre datos recolectados de sistemas de retroalimentación ciudadana operando en Boston.

Los análisis realizados en muchos de estos estudios aplican metodologías tradicionales, como análisis geoespacial y regresión. Varias propuestas han abordado la aplicación de técnicas de minería de datos, como clustering, reglas de asociación y meta heurísticas para identificar las prioridades de los ciudadanos y monitorear la percepción ciudadana, como en [15] y [16]. En [15] se realiza un análisis sobre la información del municipio de la ciudad de Bojnourd, utilizando algoritmos de clustering para determinar las necesidades ciudadanas (K-means, Fuzzy Clustering Method y Imperialist Competitive Algorithm), para finalmente comparar los resultados de cada algoritmo. Por otro lado, [16] describe el desarrollo de un conjunto de técnicas y marcos que apuntan a la gestión eficaz y eficiente de datos urbanos en entornos reales, trabajando con datos reales de la ciudad de Dublín.

El uso de técnicas de minería de datos se complementa con la aplicación de metodologías de inteligencia empresarial, como la investigación de indicadores clave de desempeño, que pueden ser utilizados como indicadores cuantitativos de la percepción de los ciudadanos sobre diferentes temas relacionados con las ciudades, como por ejemplo [17], que presenta una arquitectura integrada de minería de datos e inteligencia de negocios, basada en tecnologías abiertas, para el análisis de datos abiertos de no-emergencia adquiridos en un contexto de ciudad inteligente.

Sin embargo, a diferencia de CitymisVis, las propuestas identificadas realizan análisis offline en lugar de proveer herramientas especialmente desarrolladas para ser visuales e interactivas.

5 Conclusión

En este trabajo se presentó CitymisVis, un prototipo que ayuda a identificar problemas predominantes en un municipio. CitymisVis permite el análisis de los PQR de los ciudadanos mediante una aplicación de métodos especialmente desarrollados basados en clustering y técnicas de visualización. Estos métodos incluyen el uso de mapas de calor para representar la intensidad de reportes en diferentes áreas geográficas, la aplicación de algoritmos de clustering (basado en densidad y jerárquico) para visualizar agrupaciones de reportes a diferentes niveles de granularidad, y el uso de zoomable sunburst charts para producir representaciones navegables de datos estadísticos calculados sobre la información recolectada.

Con estos métodos, CitymisVis ofrece una valiosa ayuda para identificar las áreas más afectadas, diseñar planes logísticos para atender a las demandas ciudadanas de manera eficiente, diseñar estrategias para mejorar a tiempo y de manera proactiva la infraestructura de servicios en una comunidad, inferir cuantitativamente las relaciones de causa-efecto resultantes de las políticas públicas del municipio mediante la evaluación de acciones concretas en áreas geográficas delimitadas, y producir reportes para mantener informados a los ciudadanos mediante datos visuales y estadísticas. En consecuencia, el uso de CitymisVis conduce a una organización interna más eficiente al reducir los costos de mantenimiento. También resulta en un mejor uso de los recursos públicos, dado que conduce a tomar decisiones más inteligentes en base a la información observada. Por último, ayuda a mejorar la comunicación política y, por lo tanto, a desarrollar la comprensión entre ciudadanos y funcionarios gubernamentales.

Como parte del trabajo a futuro, se plantea incluir una dimensión temporal a los algoritmos de clustering que permita el análisis de reportes en una determinada ventana de tiempo. Esto haría posible visualizar PQR generados, por ejemplo, durante un mes específico o una semana en particular. Otra funcionalidad que se espera implementar es la posibilidad de obtener información acerca de la satisfacción del ciudadano respecto a la prestación de servicios. Esto será posible en futuras versiones de Citymis Community, donde el usuario será capaz de agregar una puntuación a la solución. Se espera, también, extender el análisis ofrecido por la herramienta al considerar otras fuentes de información, como las opiniones de los ciudadanos reflejados en las redes sociales o en comentarios de periódicos digitales. La incorporación de esta nueva información abre nuevos desafíos de investigación, como desarrollar métodos de búsqueda basados en temas, reconocimiento de entidades nombradas, análisis de sentimientos, etc.

La validación de la herramienta se ha hecho a nivel de pruebas de escritorio con respecto a la correctitud de las visualizaciones y en base a los datos almacenados. Otras validaciones experimentales se están diseando con los profesionales que desarrollaron la herramienta Citymis y en colaboración con representantes del gobierno, usuarios de la herramienta.

Agradecimientos

Esta tesis de Licenciatura en Ciencias de la Computación fue realizada en el Departamento de Ciencias e Ingeniería de la Computación de la Universidad Nacional del Sur durante el primer cuatrimestre de 2016 bajo la dirección de la Dra. Ana Gabriela Maguitman y la co-dirección del Dr. Carlos Iván Chesñevar. La misma se desarrolló en el marco del proyecto PICT-2014-0624 “Combining Context-based Search and Argumentation for delivering information in E-Gov Infrastructure”. Agradezco a la Dra. Elsa Estevez por compartir su conocimiento, al Lic. Marcos Malamud de Mismatica por sus sugerencias y por facilitarnos el acceso a los datos recolectados por Citymis y a la Municipalidad de San Isidro por autorizar el uso de estos datos.

Referencias

1. Oliveira, G.H.M., Welch, E.W.: Social media use in local government: Linkage of technology, task, and organizational context. Government Information Quarterly **30**(4) (2013) 397–405
2. Hong, H.: Government websites and social media’s influence on government-public relationships. Public Relations Review **39**(4) (2013) 346–356
3. Steibel, F., Estevez, E.: Designing web 2.0 tools for online public consultation. In: Impact of Information Society Research in the Global South. Springer (2015) 243–263
4. Fedotova, O., Teixeira, L., Alvelos, H.: E-participation in portugal: evaluation of government electronic platforms. Procedia Technology **5** (2012) 152–161
5. Desouza, K.C., Bhagwatwar, A.: Citizen apps to solve complex urban problems. Journal of Urban Technology **19**(3) (2012) 107–136
6. Sun, P.L., Ku, C.Y., Shih, D.H.: An implementation framework for e-government 2.0. Telematics and Informatics **32**(3) (2015) 504–520

7. Gamboa, F., Malamud, M.A., Maguitman, A.G., Chesñevar, C.I.: Citymis optree: Intelligent citizen management using sentiment analysis and opinion trees. In: Proceedings of the 9th International Conference on Theory and Practice of Electronic Governance, ACM (2016) 250–253
8. Rodden, K.: Zoomable sunburst with updating data. http://bl.ocks.org/kerryrodden/477c1bfb081b783f80ad.
9. Chakravarti, B., Venugopal, M.: Citizen centric service delivery through e-governance portal. A White Paper published by National Institute for Smart Government Hyderabad, India (2008)
10. Lee, D., Felix, J.R.A., He, S., Offenhuber, D., Ratti, C.: Cityeye: Real-time visual dashboard for managing urban services and citizen feedback loops. In: Proc. 14th Int. Conf. on Comp. in Urban Plann. & Urban Manag. CUPUM 2015. (2015)
11. Dirks, S., Gurdgiev, C., Keeling, M.: Smarter cities for smarter growth: How cities can optimize their systems for the talent-based economy. (2010)
12. Dahiya, D., Mathew, S.K.: Review of strategic alignment, ict infrastructure capability and e-governance impact. (2013)
13. Clark, B.Y., Brudney, J.L., Jang, S.G.: Coproduction of government services and the new information technology: Investigating the distributional biases. Public Administration Review **73**(5) (2013) 687–701
14. Offenhuber, D.: Infrastructure legibilitya comparative analysis of open311-based citizen feedback systems. Cambridge Journal of Regions, Economy and Society **8**(1) (2014) 93–112
15. Ghodousi, M., Alesheikh, A.A., Saeidian, B.: Analyzing public participant data to evaluate citizen satisfaction and to prioritize their needs via k-means, fcm and ica. Cities **55** (2016) 70–81
16. Panagiotou, N., Zygouras, N., Katakis, I., Gunopulos, D., Zacheilas, N., Boutsis, I., Kalogeraki, V., Lynch, S., OBrien, B.: Intelligent urban data monitoring for smart cities. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer (2016) 177–192
17. Cagliero, L., Cerquitelli, T., Chiusano, S., Garino, P., Nardone, M., Pralio, B., Venturini, L.: Monitoring the citizens'perception on urban security in smart city environments. In: Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on, IEEE (2015) 112–116
18. Bostock, M.: D3. data-driven documents. https://d3js.org/.
19. amCharts: Javascript charts & maps - amcharts. https://www.amcharts.com/.
20. amCharts: Simple pie chart - amcharts. https://www.amcharts.com/demos/simple-pie-chart/.
21. Mapbox.: https://www.mapbox.com/.
22. Mapbox: Create a heatmap from points. https://www.mapbox.com/mapbox-gl-js/example/heatmap/.
23. Mapbox: Markercluster with mapbox marker data. https://www.mapbox.com/mapbox.js/example/v1.0.0/markercluster-with-mapbox-data/.
24. Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., Schaub, T.: Geojson. http://geojson.org/.
25. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd. Volume 96. (1996) 226–231
26. Schubert, E., Zimek, A.: Elki data mining framework. https://elki-project.github.io/.
27. Mapbox: Clustering millions of points on a map with supercluster. https://blog.mapbox.com/clustering-millions-of-points-on-a-map-with-supercluster-272046ec5c97.

Apéndice A Conjunto de Datos

El conjunto de datos utilizados para el desarrollo de la tesis fue provisto por Mismatica, de su aplicación Citymis. Cada entrada en la base de datos está compuesta por los siguientes campos:

- ID: identificador de la entrada en el conjunto de datos.
- ID_Servicio: un número que identifica a un servicio específico. En particular, cada reporte provisto dentro del dataset corresponde a uno de los siguientes servicios: Luminaria, Arbolado Urbano, Higiene Urbana, Ingeniería vial, Señalamiento vial, Bromatología, Parques y Plazas, y Semáforos. Es destacable mencionar que Citymis provee una gran variedad de servicios y subcategorías asociadas, más allá de las nombradas.
- Categoría: corresponde a la subcategoría dentro de un determinado servicio.
- Comentario: es el comentario presentado por el ciudadano al realizar el reclamo.
- Calle y Altura: determinan la ubicación en la ciudad del reporte.
- Latitud y Longitud: las coordenadas geográficas en donde se encuentra la problemática reportada por el ciudadano.

Un ejemplo de una entrada en la base de datos provista podría ser:

ID	602159
ID_Servicio	1
Categoría	Apagada
Comentario	Está siempre apagada
Calle	CUYO
Altura	2051
Latitud	-34.495271,410000
Longitud	-58.526526,340000

En la tabla anterior, ID_Servicio 1 corresponde al servicio de Luminaria, siendo “Apagada” un problema o subcategoría que puede presentarse para ese servicio. Por otro lado, el reclamo está ubicado en Cuyo 2051, siendo la latitud y longitud presentadas las asociadas a esa dirección dentro de la ciudad. Además, el ciudadano que realizó el reclamo agregó un comentario diciendo “Está siempre apagada” para agregar información respecto al problema. Finalmente, este nuevo reclamo tomó el identificador 602159 dentro de la base de datos. Los datos de mayor valor para el desarrollo de la tesis corresponden a las coordenadas geográficas de los reportes y cada servicio y subcategoría asociada, dado que se utilizarán tanto como para ejecutar un algoritmo de clustering sobre estos datos como para visualizar cada reclamo en mapas de la ciudad. Al momento de visualizar los reclamos, también se hará uso de los comentarios asociados, a fin de que puedan observarse las opiniones ciudadanas sobre ese problema en particular, dado que un mismo problema puede tener varios reclamos asociados. Además, se utilizará la totalidad de los reportes para determinar los pesos o porcentajes de cada servicio dentro del total, así como también de cada subcategoría dentro de un servicio determinado.

Apéndice B Herramientas utilizadas

La página web de CitymisVis ofrece diversas funcionalidades relacionados con la visualización de información sobre mapas de la zona y datos estadísticos. Para las funciones provistas por el front end, con las cuales interacciona el usuario de la página web, se hizo uso de HTML, CSS, Javascript y también el framework Bootstrap. Las funcionalidades utilizan datos provistos por el servidor, desarrollado en PHP, que se encuentra encargado de la manipulación de la información asociada a los reportes ciudadanos. Se utilizó MySQL para el uso de la base de datos.

Para los gráficos que acompañan a datos estadísticos se utilizaron las implementaciones de zoomable sunburst wiht updating data y Simple pie chart, de d3 [18] y amCharts respectivamente [19]. Un sunburst permite clasificar jerárquicamente una información determinada que en forma conjunta representa una única idea, visualizándola como una circunferencia. El centro del gráfico corresponde a la única idea, que puede ir dividiéndose en secciones a su alrededor hasta llegar a las subsecciones del exterior del gráfico. El área que le corresponde a cada sección depende del valor que se le asigne. La implementación elegida de este sunburst permite realizar zoom en cualquier sección que se cliquee, pudiendo disminuir el zoom al hacer clic en el centro del círculo. Por otro lado, la implementación de gráfico de torta que provee Simple pie chart [20] permite hacer clic sobre cada porción, acción que provoca una pequeña animación que separa el fragmento seleccionado.

Para visualizar todos los mapas de la región se utilizó Mapbox [21], una plataforma que se especializa en la cartografía digital. Mapbox no sólo otorga la visualización de mapas y la posibilidad de diseñarlos de manera arbitraria, sino también gran variedad de facilidades para la visualización de diferentes objetos sobre los mapas que provee. En particular, para el desarrollo de CitymisVis se usaron principalmente tres funcionalidades:

- La visualización de los mapas de San Isidro, donde se concentran los reportes que son analizados.
- Heatmap [22], para la representación del mapa de calor.
- Markercluster [23] para la visualización de las agrupaciones de reportes.

Para representar los objetos sobre el mapa creado (figuras geométricas o un marcador sobre una ubicación), las funciones de heatmap y markercluster necesitan antes obtener información geográfica sobre estos objetos (y, opcionalmente, otros datos como un título). Para proveer esta información, CitymisVis hace uso de archivos GeoJSON [24] al permitir el intercambio de datos de manera rápida, ligera y sencilla. GeoJSON es un formato estándar para la codificación de un gran número de estructuras de datos geográficos; admite varios tipos geométricos (como Point, LineString y Polygon) y la posibilidad de agregarles información adicional (convirtiéndolos en objetos Feature).

Para la funcionalidad de visualización de clusters, CitymisVis hace uso del algoritmo de clustering DBSCAN [25], usando la implementación provista por el entorno ELKI [26]. ELKI, siglas para Environment for Developing KDD-Applications Supported by Index-Structures, es un software de código abierto escrito en Java para minería de datos. ELKI provee algoritmos con altos niveles de performance y escalabilidad, pro-

porcionando importantes ganancias en rendimiento, razón por la cual ha sido elegido sobre otros entornos o implementaciones de DBSCAN.

Para una observación dinámica e interactiva de subclusters, CitymisVis usa el algoritmo hierarchical greedy clustering [27] usando la implementación que provee Mapbox, tanto para su funcionalidad de heatmap como de Markercluster.

B.1 DBSCAN

De acuerdo con el algoritmo density-based spatial clustering of applications with noise (conocido como DBSCAN), los clusters se definen como áreas de mayor densidad respecto al resto del conjunto de datos. Un cluster encontrado con DBSCAN satisface dos propiedades: todos los puntos de un mismo cluster están densamente conectados; y, si un punto es densamente alcanzable desde cualquier otro punto en el cluster, entonces también forma parte del cluster.

El algoritmo DBSCAN requiere de dos parámetros para su ejecución: ε (épsilon), el valor de distancia requerida para determinar la cercanía entre puntos; y minPts, la mínima cantidad de puntos que son requeridos para que una región se considere densa.

El algoritmo, a grandes rasgos, funciona de la siguiente manera:

1. Se comienza con un punto p arbitrario que no haya sido visitado.
2. Se computa el vecindario de puntos de p en un radio ε de distancia.
3. Si el vecindario contiene los puntos suficientes (minPts), se crea un nuevo cluster. De forma contraria, el punto se considera outlier. Sin embargo, este punto outlier puede luego encontrarse en el vecindario de otro punto y ser considerado parte de otro cluster.
4. Si un punto se incluye en la parte densa de un cluster, su ε-vecindario también es parte de ese cluster. De este modo, todos los puntos encontrados en la ε-vecindad son agregados, así como también sus propios ε-vecindarios cuando ellos también son densos. Este proceso continúa hasta que el cluster densamente conectado es encontrado en su totalidad. Luego, se vuelve al paso 1 hasta visitar todos los puntos.

La implementación provista por ELKI también permite configurar otras opciones además del minPts y épsilon, como la función de distancia o el modelo geográfico utilizado por el algoritmo.

B.2 Hierarchical greedy clustering

Mapbox utiliza este algoritmo para lograr la visualización de agrupaciones de la manera más eficiente posible, dado que su rendimiento es lo suficientemente bueno como para ejecutarse dinámicamente incluso con un gran número de puntos.

El algoritmo greedy clustering funciona, a grandes rasgos, de la siguiente manera:

1. Se selecciona un punto del conjunto de datos.
2. Se encuentran todos los puntos cercanos al elegido dado un determinado radio.
3. Con los puntos encontrados, se crea un nuevo cluster.

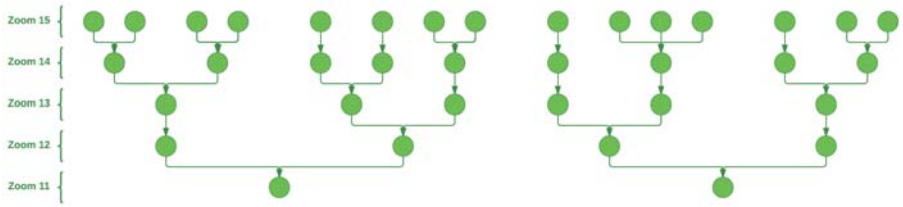

Figura 12: Ejemplo de hierarchical greedy clustering.

4. Se elige un nuevo punto que no forma parte de ningún cluster, y se repiten los pasos hasta haber visitado todos los puntos del conjunto de datos.

Este algoritmo sirve para encontrar agrupamientos de puntos a un determinado nivel de zoom en un mapa, pero sería muy costoso hacerlo para todos los niveles de zoom permitidos. Por ejemplo, si los niveles de zoom van de 0 a 15, se debería procesar todo el conjunto de datos dieciséis veces, uno por cada nivel. Además, realizar clustering en los niveles inferiores de zoom tardaría demasiado dado que cada agrupación tendría que alojar a un número exponencialmente creciente de puntos.

Este problema puede evitarse reutilizando evaluaciones previas para calcular las nuevas agrupaciones. Luego de realizar clustering sobre un nivel de zoom determinado (por ejemplo, el zoom 15), se pueden agrupar los clusters resultantes para determinar los clusters del zoom inmediato inferior (siguiendo el ejemplo, del zoom 14), y así sucesivamente (del 14 al 13, del 13 al 12, etc.). Dado que el número de puntos a analizar decrece exponencialmente en cada paso, como puede verse en Figura 12, se puede realizar clustering en todos los niveles más eficientemente. A este enfoque se lo conoce como hierarchical greedy clustering siendo el que finalmente utiliza Mapbox.

Apéndice C Organización de Archivos

Para las diferentes secciones de la página web es necesario recuperar archivos previamente generados por el servidor, para finalmente visualizar la información asociada a los reclamos de la manera adecuada según la sección a la que corresponda. La generación de estos archivos queda a cargo de servidor, que tiene diferentes rutinas en las cuales recupera información de la base de datos y genera los archivos GeoJSON o JSON según sea necesario.

Todos los archivos generados están dentro de la carpeta files en el servidor. Dentro de este directorio, hay otras tres carpetas: clusters, coordinates y geojson, cuyo contenido y utilidad se describirá más adelante. Además, contiene cuatro archivos JSON:

- clustering_config_values.json, que guarda los valores para los parámetros de entrada al algoritmo DBSCAN. Este archivo define la escala para la cantidad de reportes que se analizarán junto a los valores para épsilon y minPts que tiene asociado, y es utilizado exclusivamente en la sección dedicada a la visualización de clusters. Un ejemplo de este archivo puede verse en el Cuadro 1.

- layers.json, que contiene la escala de valores para el heatmap, definiendo los rangos de reportes y el color que se utiliza para esa concentración de reclamos en el mapa de calor, como puede observarse en el Cuadro 2. Se hace uso de este archivo para la generación del mapa de calor, así como también para las referencias dentro del mapa.
- services.json, define cada uno de los servicios con información asociada, como nombre, el color e ícono que se utilizarán para él en la página y las subcategorías que lo componen, además del número que lo identifica. Este archivo se utiliza en todas las secciones de la página web. Un extracto puede verse en el Cuadro 3.
- zoomable.json, archivo JSON que se utiliza para la visualización del gráfico zoomable sunburst with updating data, que acompaña información estadística relacionada con la cantidad de reclamos generados.

Cuadro 1: Valores para DBSCAN

```
{
  "250": [
          {
            "eps": 500,
            "minPts": 3
          },
          {
            "eps": 700,
            "minPts": 7
          }
        ],
  "500": [
          {
            "eps": 700,
            "minPts": 9
          }
        ]
}
```

En la tabla puede verse una parte de layers.json, que determina que aquellas agrupaciones que tengan una cantidad de reportes desde cero en adelante usarán el color determinado por rgba(102, 204, 102, 0.4); así mismo, determina que las agrupaciones que contengan 125 reportes o más usarán el color rgba(255, 204, 102, 0.6).

Para generar los archivos que están dentro de la carpeta coordinates (ver Figura 13), se ejecuta una rutina del servidor que se encarga de recuperar datos de los reclamos

Cuadro 2: Definición de grados

```
[
  [
    0,
    "rgba(102, 204, 102, 0.4)"
  ],
  [
    125,
    "rgba(255, 204, 102, 0.6)"
  ]
]
```

En la tabla puede verse una parte de layers.json, que determina que aquellas agrupaciones que tengan una cantidad de reportes desde cero en adelante usarán el color determinado por rgba(102, 204, 102, 0.4); así mismo, determina que las agrupaciones que contengan 125 reportes o más usarán el color rgba(255, 204, 102, 0.6).

de la base de datos referente a las coordenadas geográficas donde están ubicados los reportes y a qué subcategoría y servicio pertenece cada uno, volcando esta información en archivos txt. Hay uno de estos archivos por cada servicio, y uno más para la totalidad de los reclamos, identificándose cada archivo con el nombre coordinates_idService.txt, siendo idService el número que identifica al servicio (con el valor cero para señalar el que guarda todos los reclamos). El contenido de estos txt seguirá el siguiente formato:

Latitud Longitud # idService | Subcategoría

donde Latitud y Longitud corresponden a las coordenadas geográficas donde está ubicado el reclamo; el numeral se utiliza para saber que lo que sigue es información adicional; idService es el identificador del servicio y Subcategoría es la que corresponde al reclamo en cuestión. Un ejemplo podría ser:

-34.473676990000 -58.540745970000 # 1 | Encendida de Día

De este modo, los dos primeros valores corresponden a la latitud y longitud de un reclamo perteneciente al servicio con identificador 1 (uno), ubicado dentro de la subcategoría “Encendida de Día”.

Estos archivos se utilizan como entrada en el algoritmo DBSCAN para la generación de clusters —por lo que requieren el formato descrito—, junto a los parámetros épsilon y minPts. La rutina que utiliza estos archivos es la encargada de correr el DBSCAN para cada servicio y cada posible valor de los parámetros mencionados. La ejecución del algoritmo genera una serie de archivos txt que se corresponden a cada cluster encontrado, y uno extra con los outliers. Estos archivos se guardan en una carpeta con el formato de nombre: clusters_epsN_minPtsM, donde N se corresponde con el valor de épsilon y M de minPts usado para esa instancia. A su vez, todas las carpetas que se

Cuadro 3: Información del servicio

```
{ "8" :
  {
    "nombre": "Semáforos",
    "color": "#FF6633",
    "icon": "icon-trafficlight",
    "categorias":
      [
        "No funciona indicador peatonal",
        "No funciona luz amarilla",
        "No funciona luz roja",
        "No funciona luz verde",
        "Semáforo chocado",
        "Semáforo no funciona",
        "Semáforo roto",
        "Solicitud de nuevo semáforo"
      ]
  }
}
```

En este fragmento de services.json, se muestra el modo en que se dispone la información asociada al servicio con identificador igual a 8: el nombre, el color e ícono que se usará para él en la página, y las subcategorías que lo constituyen.

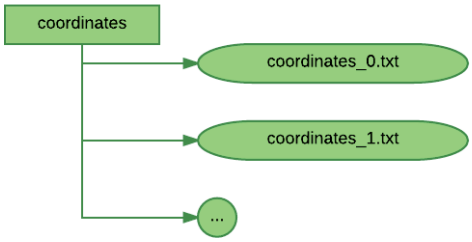

Figura 13: Organización de la carpeta coordinates.

creen para un mismo conjunto de datos inicial (dado que el algoritmo se corre para un determinado conjunto de reclamos con diferentes valores en los parámetros de entrada), estarán dentro del directorio idServicio, que es el número que identifica al servicio, siendo el cero el que identifica a la totalidad de los servicios. Asimismo, todas estas carpetas estarán ubicadas dentro de clusters. Esto puede verse en la Figura 14.

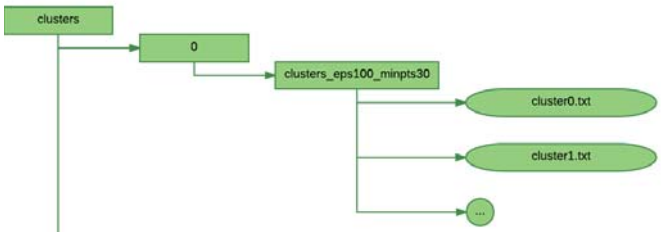

Figura 14: Organización de la carpeta clusters.

En el ejemplo de la figura, para todos los servicios (carpeta con identificador cero) se utilizan el algoritmo DBSCAN épsilon=100 y minPts=30. De este modo, se crea dentro del directorio clusters/0 la carpeta clusters_eps100_minPts30 que contendrán los archivos que guardan los reportes que conforman las distintas agrupaciones encontradas por el algoritmo (cluster0_txt, cluster1.txt, etc.).

Otra rutina hace uso de estos archivos generados por el algoritmo DBSCAN de ELKI para generar los archivos GeoJSON que se usan posteriormente para la visualización de las agrupaciones encontradas, formateando los datos de los txt para que cumplan con el formato requerido por GeoJSON. La organización de estos archivos dentro del directorio sigue la misma forma que lo descrito anteriormente, con la única diferencia de que la carpeta cluster —que contiene archivos GeoJSON— se encuentra dentro de una carpeta llamada geojson, como puede observarse en la Figura 16. Esta carpeta también

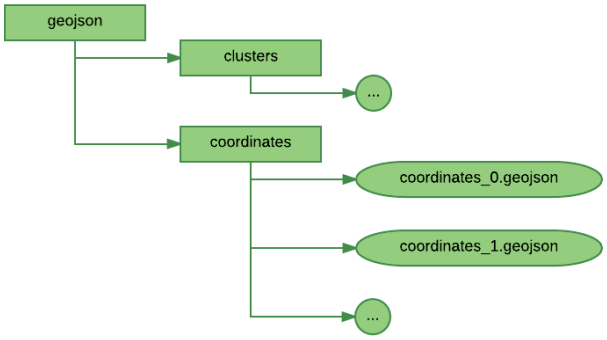

Figura 15: Organización de la carpeta geojson/coordinates.

Las coordenadas geográficas en formato geojson de los diferentes servicios (coordinates_0.geojson, coordinates_1.geojson, etc.) se encuentran dentro de la carpeta coordinates, a su vez dentro de geojson.

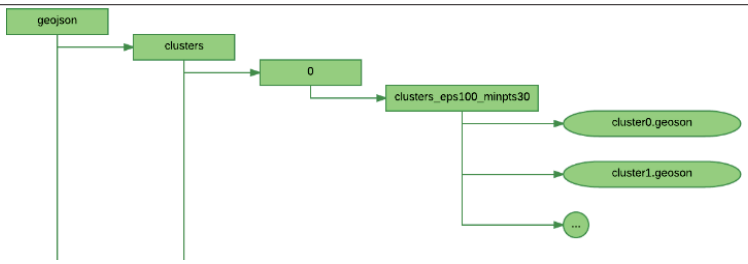

Figura 16: Organización de la carpeta geojson/clusters.

La organización de los archivos sigue el mismo formato que lo mostrado en la Figura 14, con el uso del directorio clusters/0 y la carpeta que contiene los resultados de correr el algoritmo con eps=100 y minPts=30. Sin embargo, los archivos siguen ahora el formato especificado por geojson (cluster0.geojson, cluster1.geojson, etc.), y todas las carpetas están a su vez en la carpeta geojson.

contiene los archivos GeoJSON que se corresponden con la información que guardan los archivos dentro del directorio coordinates, siendo otra rutina del servidor la encargada de generarlos y ubicarlos dentro de una carpeta del mismo nombre incluida en geojson, utilizando el mismo formato para el nombre de los archivos (ver Figura 13).

Finalmente, la organización de los archivos utilizados puede verse resumida de la Figura 17.

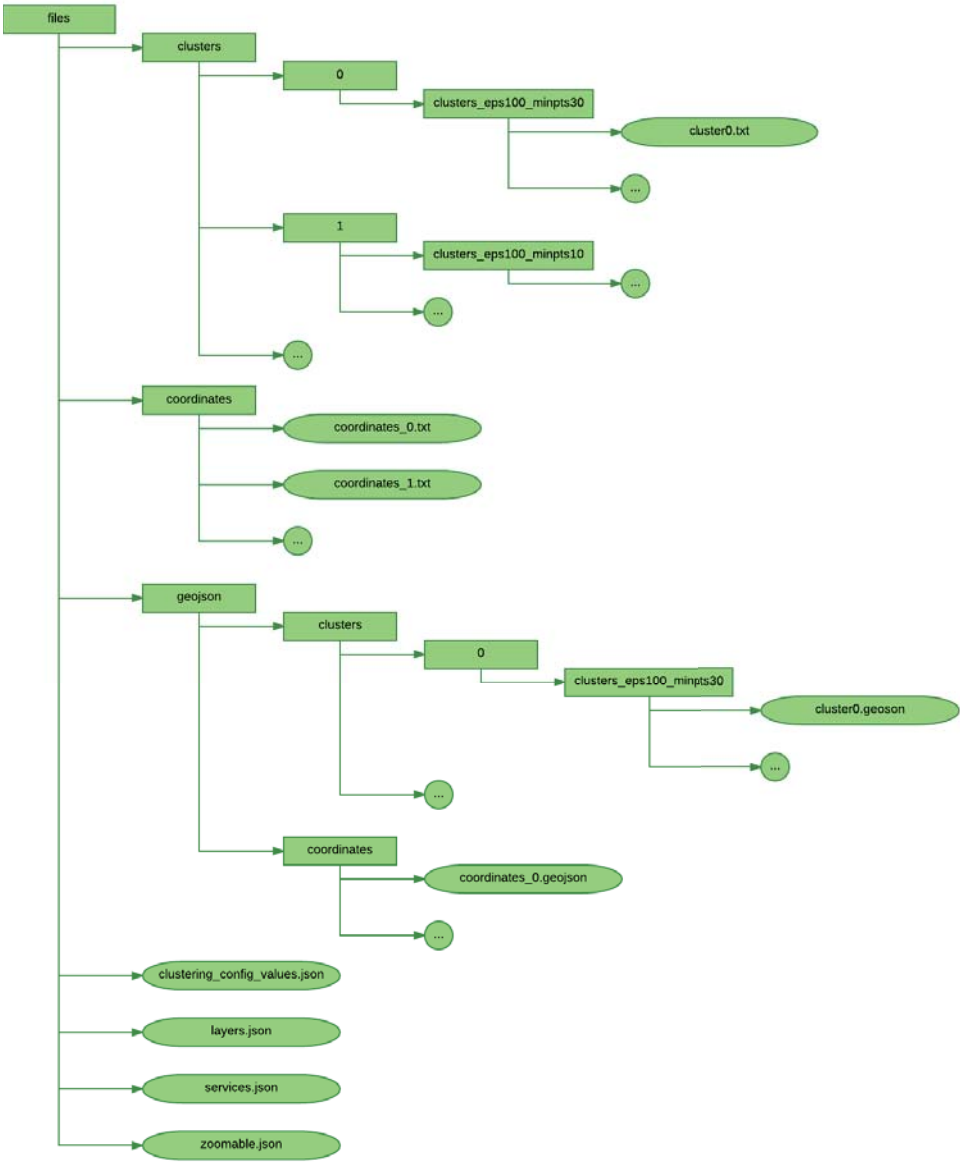

Figura 17: Organizacin de la carpeta files.