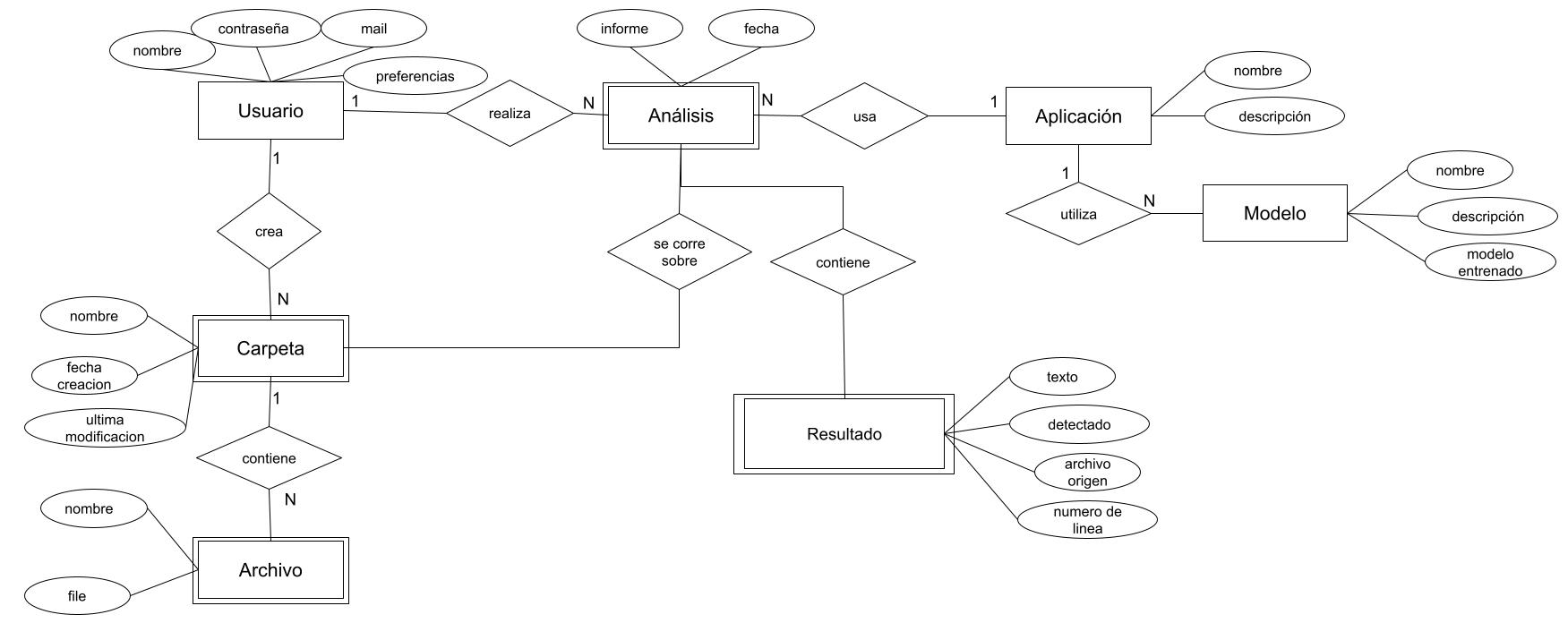
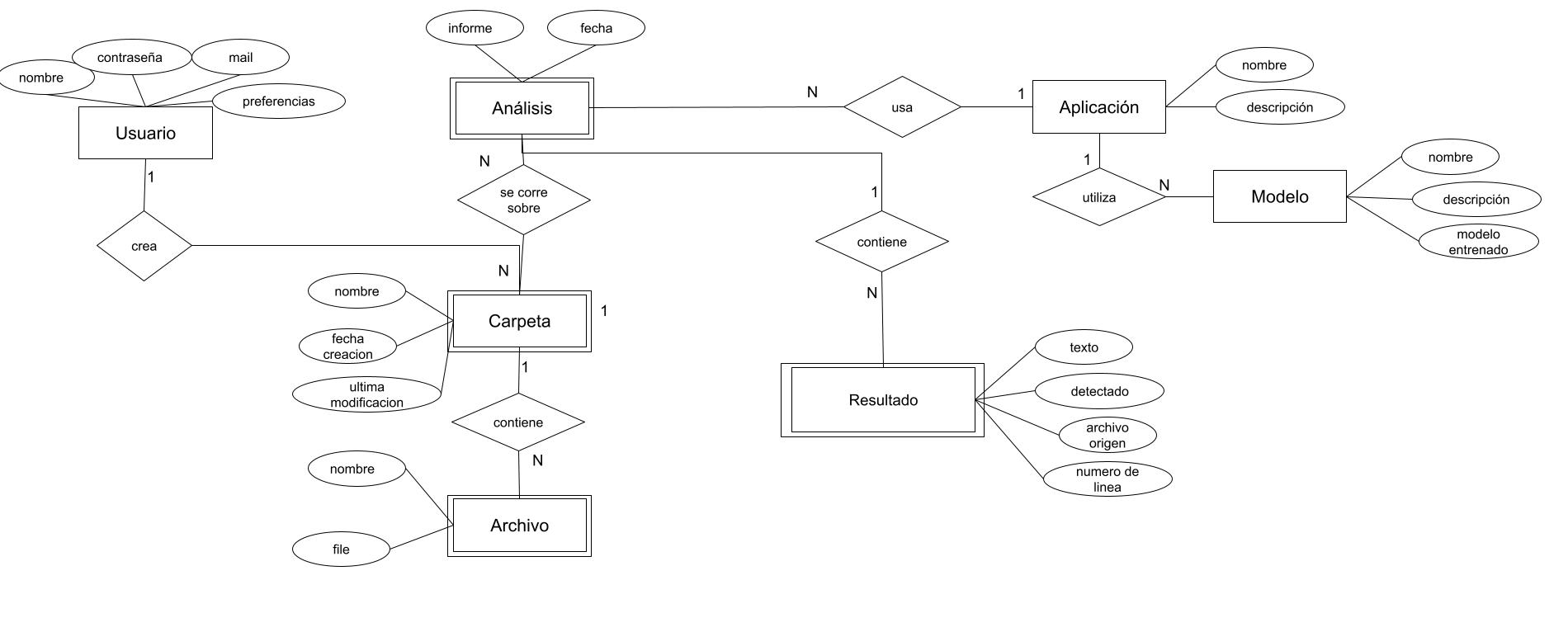
# Notas del desarrollo de la aplicación web

Cambio de los requerimientos en la funcionalidad de carpetas: se pidió que el usuario pueda subir sus archivos a analizar a una carpeta y que después los resultados de este análisis están en una sección resultados.

Creamos el Diagrama de entidad relación para definir la base de datos que se va a usar en el django



Se modificó el diagrama ER para evitar redundancias y seguir las convenciones de representación (lectura de arriba a abajo y de izquierda a derecha).

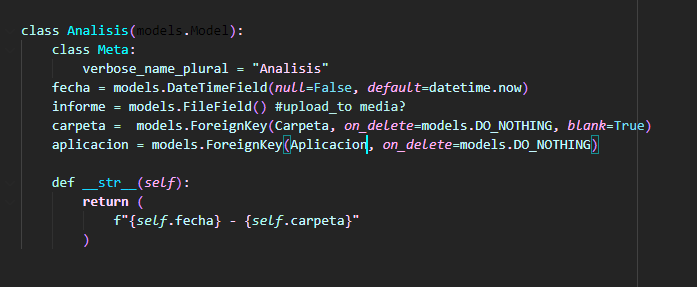


Creé el proyecto django, definí en models.py la estructura básica de la BD, cree la app para el manejo de usuarios, el manejo de urls, y definí los templetes a crear.

Empecé a crear los templates y los estilos correspondientes, hubo algunas dificultades con el manejo de los archivos estáticos pero luego de consultar la documentación de Django se pudo arreglar.

En esta primera versión el css es muy básico deberíamos cambiarlo para que se parezca más a los diseños que creamos previamente. Figma permite descargar el css de cada elemento pero es medio desprolijo, ver si podemos extraer lo importante de cada uno.

Se definió en primera instancia que el análisis tenga una aplicación asociada y no un modelo. Es decir que la aplicación use los modelos predeterminados que tenga pero no se pueden elegir.



Para acercarse más a los requerimientos se redefinió esta clase para que contenga modelos no aplicaciones(el modelo tiene la referencia a la app)

El proceso de desarrollo es: implementación de funcionalidad, testeo unitario, arreglos, testeo integrado, arreglos y observaciones.

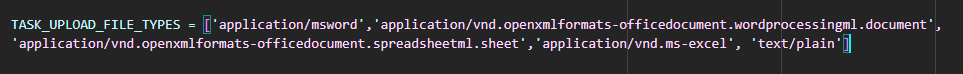
\_\_\_

Problemas: validación de la extensión de un archivo en el forms es complicada (principalmente por la seleccion de multiples archivos que separa la tabla de modelos Archivo de el form usado para subir varios archivos que luego se convierten en objetos archivo)

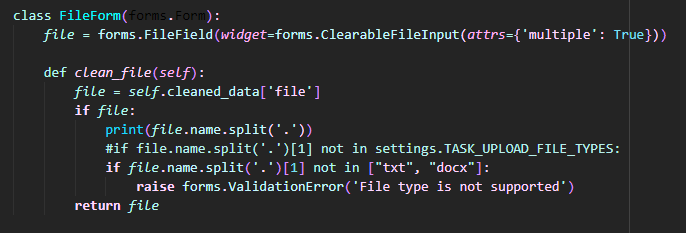
Una posible solución es dejar que la persona suba cualquier cosa y después en el back se ignoren los archivos que tengan un tipo X.

Por alguna razón esto no funciona:

en settings.py



en forms.py



El problema es que la función clean deja pasar a todos los archivos seleccionados si se elige al menos un archivo correcto

SOLUCIONADO: Se hace la validación en el form y en el momento de hacer el procesamiento del análisis. Si quedo un archivo incorrecto aparece un warning.

-Además los archivos no se están levantando con encoding ‘utf-8’, en la documentación de django el filefield se abre haciendo filefield.open(‘r’) y no tiene un atributo encoding.

SOLUCIONADO ( **with** open(archivo.arch.path, "r", *encoding* = 'UTF-8') **as** f:)

-Los archivos subidos no se están guardando en una carpeta aparte.

SOLUCIONADO: ahora se guardan en /archivo

- Solo se están procesando archivos .txt (ver manejo de .doc o .pdf(que se lea) o .zip) nota: si se permiten .zip o .pdf, se debería hacer una validación al hacer la lectura y avisar al usuario si hay un archivo que no puede ser procesado. (el pdf es una imagen/ el zip contiene archivos de tipos no permitidos)

LISTO el manejo de los distintos tipos de archivo. Se tuvo que modularizar el procesar un archivo para poder extraer del zip si fuera necesario y trabajar con el archivo abierto (dado a que los archivos del zip no tienen un path) Y además permitiendo que luego se muestran en los resultados completos a que archivo dentro del zip pertenece cada frase.

-



Los resultados se deberían ver al revés, del más nuevo al más viejo. SOLUCIONADO

**Opciones para incorporar los modelos al django:**

CELERY con RabbitMQ

Desventaja: tiempo que tardan los modelos en cargarse.

Ventaja: no se tiene que mantener un servidor funcionando.

[First Steps with Celery — Celery 5.3.3 documentation (celeryq.dev)](https://docs.celeryq.dev/en/stable/getting-started/first-steps-with-celery.html#rabbitmq)

[RabbitMQ: easy to use, flexible messaging and streaming — RabbitMQ](https://www.rabbitmq.com/)

[First steps with Django — Celery 5.3.3 documentation (celeryq.dev)](https://docs.celeryq.dev/en/stable/django/first-steps-with-django.html)

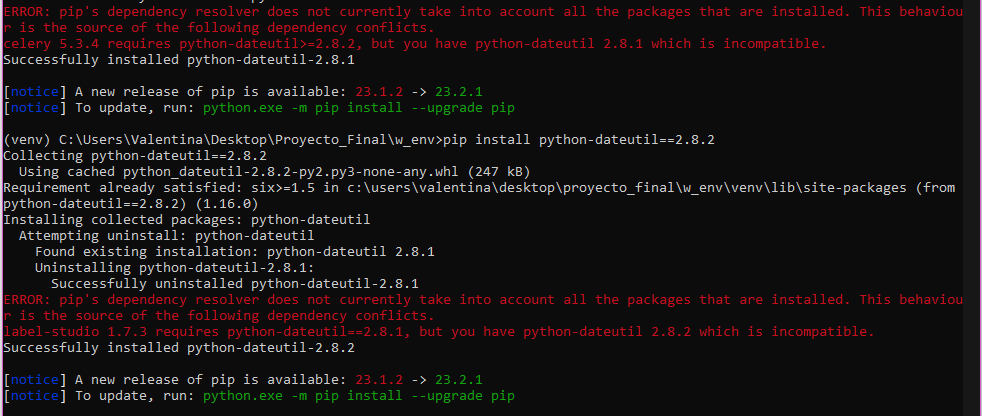
\*\* chequear

[Building Progress Bars for the Web with Django and Celery | Build with Django](https://buildwithdjango.com/blog/post/celery-progress-bars/)

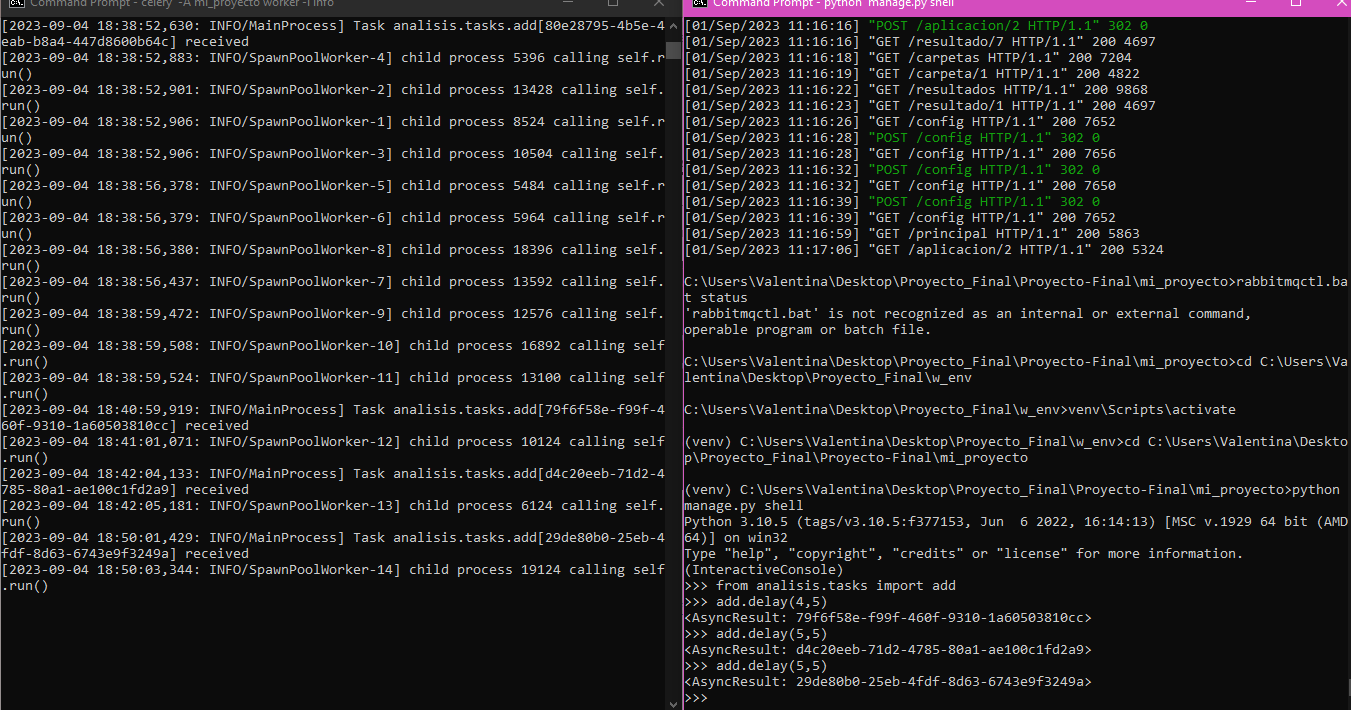
video simple:

[Learn Django Celery with RabbitMQ - Install and create new celery instance, Run a simple task Part 1 - YouTube](https://www.youtube.com/watch?v=fBfzE0yk97k&list=PLOLrQ9Pn6caz-6WpcBYxV84g9gwptoN20)

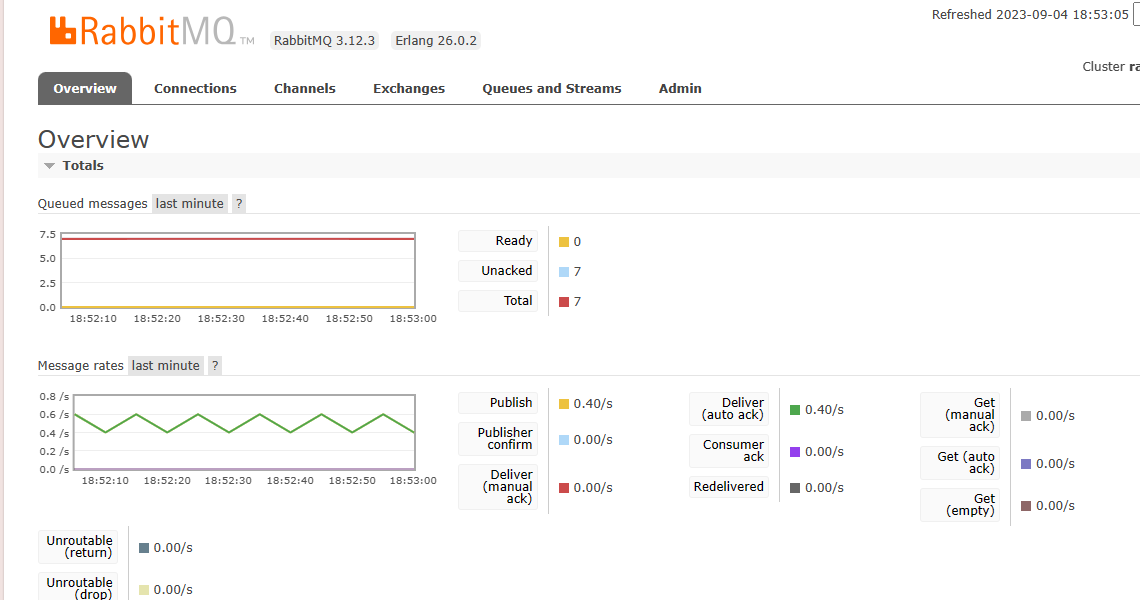
nota: hay un problema de dependencias con label-studio y celery: ( ponerlos en dos venvs distintos)



además, luego de instalar celery y rabbitMQ, crear un task simple y el celery.py se ve que se reciben las tareas pero no se ejecutan



RabbitMQ está funcionando bien.



Aparentemente el problema puede ser windows:

[python - task receive but doesn't excute - Stack Overflow](https://stackoverflow.com/questions/62524908/task-receive-but-doesnt-excute)

Seguí la recomendación mostrada en el link pero ahora si bien me muestra lo que llega da un error en analisis.tasks.add y aun no ejecuta o muestra el resultado. Capaz hay que intentar con linux directamente.

Después de investigar y probar configuraciones, se logró reproducir correctamente la creación y la carga de la barra, tanto en Linux como en Windows.

Para evitar problemas de dependencias como los descriptos se encontró que la mejor solución es el uso de venv.

Además, para poder hacer fetch de las constantes actualizaciones de estado de manera asincrónica, se necesita el uso de Redis además de RabbitMQ y Celery. Esto es porque Rabbit solo envía los mensajes y los status, pero no se guardan.

Además había un par de configuraciones adicionales de Celery.

→ Se deberia: - mandar el formulario y llamar a la url “Procesar analisis” que sea como el “comenzar celery” que está definido en el branch de celery del proyecto. Ahí deberia estar la barra de progreso. En comenzar celery podemos llamar al método del módulo nlp. y una vez que termina llamar al resultado.

Modelos NLP

Visualización:

Informe con gráficos realizados por librerías de visualización como altair:

[python - how do I get my altair plot from views to render on my html page in Django? - Stack Overflow](https://stackoverflow.com/questions/69021750/how-do-i-get-my-altair-plot-from-views-to-render-on-my-html-page-in-django)

A lo largo del desarrollo se encontraron varios problemas con la visualización de los elementos gráficos y el armado del informe. Se exploro la idea de el cambio de librería, o de en lugar de pasar los gráficos (posibilidad de interactividad, descarga, inspección de los datos) pasar un png con el gráfico (estático) pero esto además de ser una peor opción resultó más complejo de realizar ya que la librería no implementa un método nativo para este guardado sino que se depende de una librería altair\_saver que depende de altair\_viewer y que necesita selenium con un driver de chrome específico (que trae complicaciones de compatibilidad) después de dos días de pruebas para el uso de pngs, se desistió y se decidió en lugar de intentar enviar en el atributo ‘informe’ de análisis, un html terminado que será insertado en el html ‘padre’ de resultado, se creó el modelo Grafico, Grafico\_imagen y Tabla todos elementos que se usan para crear el informe en el mismo html de ‘resultados’ y los problemas al renderizar los gráficos se solucionaron usando esta metodología, pasando grafico por grafico (siguiendo recomendaciones de la documentación de altair) y separando el template ‘resultado.html’ en resultado\_entidades.html y resultado\_clasificador.html. Permitiendo así el armado de los dos tipos de informes y seguir priorizando la extensibilidad, modificabilidad etc.

\_

Al comenzar a gestionar como sería el manejo de textos que son mensajes de whatsapp surge la duda de la necesidad de realizar cambios a la base de datos: Algunos objetos generados con un tipo de archivo específico van a tener datos como: fecha, usuario y si corresponden a un archivo adjunto (en cuyo caso no tendrían nada en 'detectado') Es mejor, agregar esos campos a Resultado aunque puedan o no estar vacíos? o crear otro modelo (Mensaje\_wpp por ejemplo) que referencie a un objeto resultado y tenga esos campos adicionales.

Agregar campos adicionales al modelo Resultado: (mejor performance)

Pros:

No necesitas crear un nuevo modelo y gestionar las relaciones entre ellos. Podes recuperar todos los datos que necesitas con una sola consulta a la base de datos.

Contras:

Campos vacíos.

Flexibilidad: Si en el futuro necesitas agregar más campos específicos para diferentes tipos de archivos, tu modelo Resultado puede volverse cada vez más grande y difícil de manejar.

Crear un nuevo modelo que referencia a Resultado:(uso más eficiente del espacio)s

Pros:

Especialización: Puedes tener modelos separados que estén optimizados para diferentes tipos de archivos.

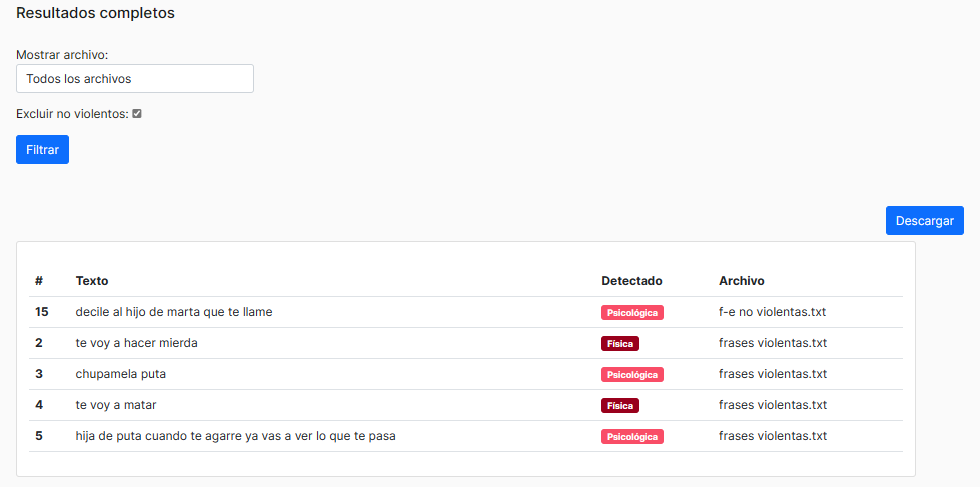
Si en el futuro necesitas manejar más tipos de archivos con campos únicos, puedes simplemente crear nuevos modelos para ellos sin tener que modificar el modelo Resultado.

Contras:

Complejidad: Necesitas gestionar múltiples modelos y sus relaciones, lo que puede complicar tu código.

Rendimiento: Puede que necesites realizar múltiples consultas a la base de datos para recuperar todos los datos que necesitas.

Otra cosa que se puede hacer es agregar un campo extra que sea ’flexible’ como ‘datos extra’. Pero eso puede traer problemas de claridad en el código, y de todas formas puede ser un campo vacío.



\*se puede poner siempre otras dos columnas con fecha y usuario.

Agregar opción de filtrar por usuario o por fecha.

11/5/2023, 22:21 - Valentina: ‎STK-20220611-WA0038.webp (archivo adjunto)

-> guardaría Resultado con detectado = ‘Archivo adjunto’, texto=STK-20220611-WA0038.webp

* Puedo chequear no cada línea sino las primeras líneas para ver si es un archivo de wpp y si lo es asumo que todo el archivo es así.

si se exporta sin archivos no dice (archivo adjunto) al final, dice:

17/10/2023, 17:59 - Valentina: <Multimedia omitido>

Si hay /n en los mensajes se van a incluir en el archivo así:

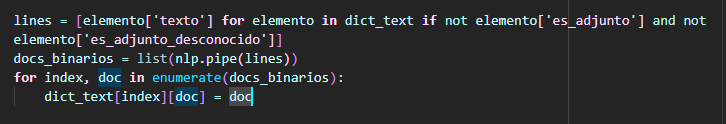
19/10/2023, 18:33 - Valentina: Hola!

Yo

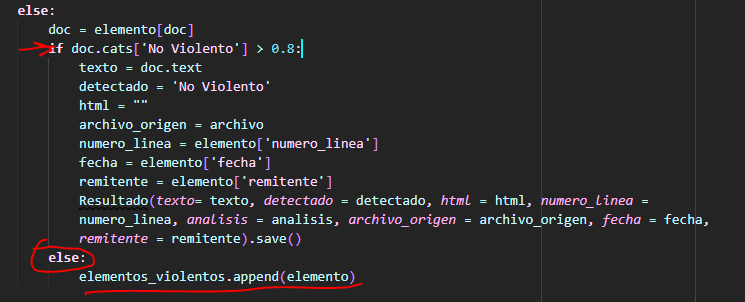
entonces si se lee línea por línea, algunas líneas del archivo no van a cumplir con el formato (tengo que chequear con la expresión regular en cada paso. Se puede definir que si un archivo es de wpp ( primera línea cumple el formato) se busca cada mensaje cuando empieza cada fecha, o directamente analizar línea por línea. Y dejar esas líneas separadas por espacios como mensajes con usuario o fecha desconocidos.

Teniendo todo esto en cuenta se redefinio la funcion procesar\_clasificador()

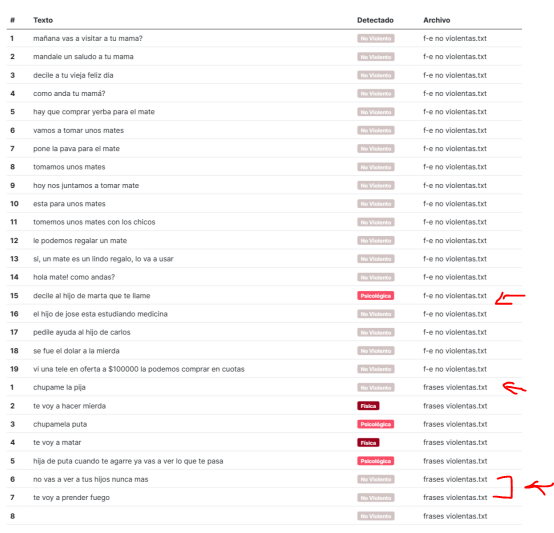
Esta funcion ahora primero abre el archivo, lo recorre y por cada linea no vacía, chequea si tiene el formato de wpp usando una expresion regular, y crea un ‘elemento’ (diccionario con los datos extraidos. Posteriormente se procesan usando nlp.pipe() las lineas de texto correspondientes. (esto no se hace elemento por elemento en un mismo for porque la libreria spacy proporciona el metodo más eficiente de usar nlp.pipe(muchas\_lineas), en lugar de for linea in muchas lineas: nlp(linea) )Como esto mantiene el orden de los resultados se agrega el doc, al ‘elemento’ (debido a que posteriormente se van a utilizar diferentes secciones del elemento segun el tipo de texto presente (archivo adjunto, violento, no violento)



\*\*NOTA: En esta iteración a modo de prueba, para evitar falsos negativos se cambio el criterio si no se encuentra un no violento con score más del 0.8 se concidera violento.

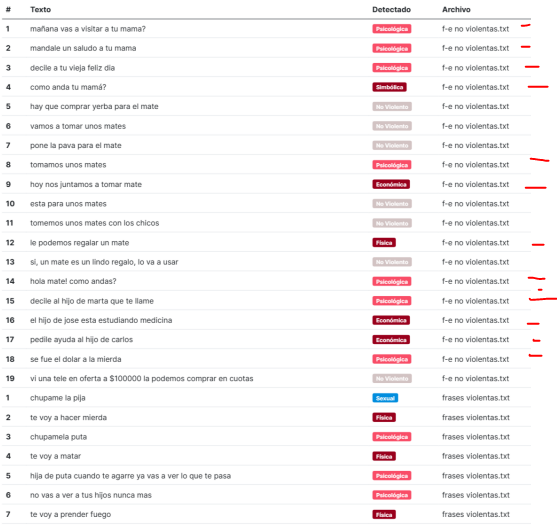


En el caso en que consideramos a los violentos los que tenian más de 0,8 Violento:



tres falsos negativos y un falso positivo

Pedir no violentos > 0.8

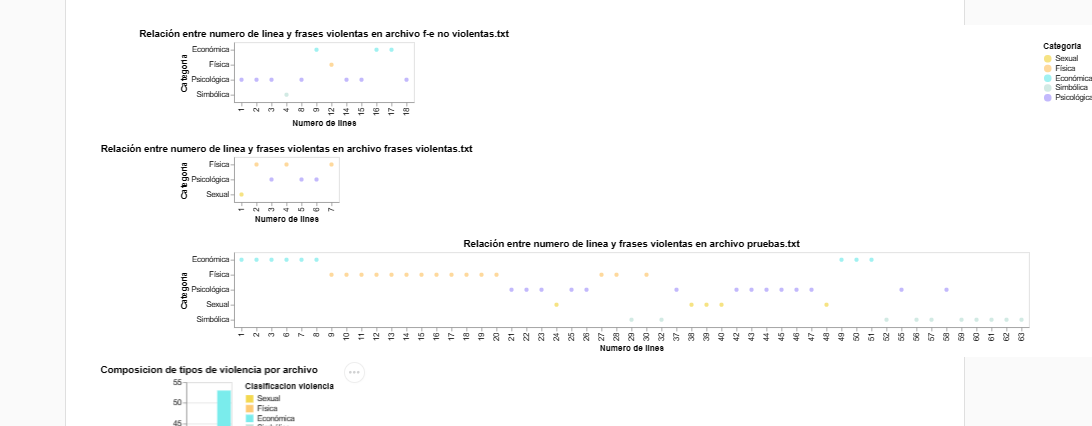


12 falsos positivos

Por lo tanto se fue por la primera opción.

El grafico se pasa del borde.

(ARREGLADO: se definio el width y el axis)

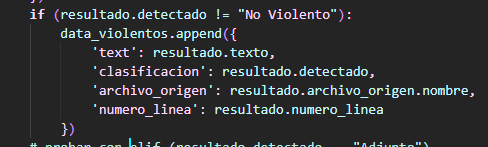


Problema:

Wordcloud de ‘violentos’ incluye los adjuntos



intente agregar and resultado.detectado!= “Adjunto” pero da error de tipo en clasificación (no puede detectar el tipo).



LISTO: casteo a string + cambios más adelante

Decidimos pasar las frases a minuscula antes de procesarlo con el clasificador de violencia porque consideramos que da mejores resultados a cambio de perder la información de las mayúsculas.