Informe de Proyecto: Report Generator

Un Sistema de Curación y Síntesis de Noticias Personalizadas

Equipo NLP

22 de enero de 2026

Resumen

Report Generator es un sistema de software que aborda el desafío de la sobrecarga informativa mediante la automatización de un flujo de trabajo completo: desde la ingesta de noticias hasta la entrega de un reporte personalizado, resumido y listo para su consumo. El sistema se nutre del canal de Telegram de **Telesur en español**, procesa los artículos para construir un corpus de conocimiento, y lo utiliza para generar informes alineados con los intereses específicos de cada usuario.

El valor diferencial del proyecto reside en la combinación de tres capacidades clave:

1. **Motor de recomendación explicable**, que no solo selecciona contenido relevante, sino que justifica sus decisiones mostrando las coincidencias temáticas y de entidades que motivaron la selección.
2. **Capacidad de síntesis dual**, que puede operar con un modelo algorítmico clásico (TextRank) o con un modelo avanzado de Deep Learning (basado en redes Pointer-Generator) para generar resúmenes de alta calidad.
3. **Arquitectura autocontenida y portable**, que utiliza una API ligera (FastAPI) y persistencia en archivos JSON, garantizando un despliegue sencillo y una alta auditabilidad.

El resultado es una herramienta funcional que transforma un flujo de noticias en bruto en inteligencia accionable, entregada en un formato profesional y portable (PDF).

1. Introducción

En el ecosistema mediático actual, el acceso a la información es ilimitado, pero la capacidad de atención del usuario es finita. La verdadera ventaja competitiva no radica en tener más datos, sino en extraer el conocimiento relevante de manera eficiente. Report Generator fue concebido para resolver este problema, actuando como un analista personal automatizado que monitorea una fuente de noticias, la filtra según los intereses del usuario y presenta los hallazgos de forma concisa.

El proyecto se distingue por ir más allá de una simple recomendación. Su propósito es entregar un producto final de alto valor: un **reporte de inteligencia** que no solo informa, sino que contextualiza y sintetiza, permitiendo al usuario asimilar los puntos clave de múltiples noticias en una fracción del tiempo que requeriría su lectura individual.

2. Metodología y Arquitectura Técnica

El sistema está organizado en un pipeline de cuatro etapas principales, cada una con componentes técnicos específicos que contribuyen al resultado final.

2.1. Etapa 1: Ingesta y Estructuración de Datos

La base del sistema es un corpus de noticias actualizado y bien estructurado. La fuente principal es el canal de Telegram de **Telesur en español**.

- **Extracción Automatizada**: Un componente de *scraping* monitorea el canal, extrae las URLs de los artículos compartidos y recopila metadatos asociados al mensaje original (fecha, vistas, reacciones).
- **Procesamiento de Contenido**: Cada URL es procesada para extraer y estructurar la información clave del artículo: título, sección, etiquetas temáticas y el cuerpo del texto completo. Se aplican filtros para eliminar ruido, como frases promocionales o enlaces no pertinentes.
- **Almacenamiento Estructurado**: Los artículos procesados se guardan en formato JSON, creando un corpus local que sirve como la única fuente de verdad para el resto del sistema. Esta arquitectura de archivos facilita la portabilidad, el versionado y la depuración.

2.2. Etapa 2: Modelado de Perfil de Usuario y Recomendación

El núcleo de la personalización reside en un motor de recomendación que entiende los intereses del usuario y los coteja con el corpus de noticias.

- **Creación de Perfil Semántico**: Los intereses del usuario, expresados a través de una consulta en lenguaje natural o mediante la selección de categorías, se transforman en un *perfil vectorial*. Este perfil no es una simple bolsa de palabras, sino una representación matemática en un espacio de alta dimensionalidad que captura las relaciones semánticas entre los conceptos de interés.
- **Vectorización del Corpus (TF-IDF)**: Todos los artículos del corpus son procesados y convertidos en vectores utilizando el algoritmo TF-IDF (Term Frequency-Inverse Document Frequency). Este método pondera la importancia de cada término no solo por su frecuencia en un artículo, sino también por su rareza en el conjunto total de documentos, otorgando mayor peso a las palabras que son verdaderamente distintivas de un tema.
- **Matching por Similitud de Coseno**: El sistema calcula la similitud del coseno entre el vector del perfil del usuario y el vector de cada artículo. Este cálculo produce una puntuación de relevancia que permite ordenar los artículos de más a menos pertinentes.
- **Recomendación Explicable (XAI)**: Una de las contribuciones más significativas del proyecto es su capacidad de justificar las recomendaciones. Junto con la puntuación de relevancia, el sistema devuelve las *categorías* y *entidades* (personas, lugares, organizaciones) que coinciden entre el perfil del usuario y el artículo, ofreciendo una transparencia que fomenta la confianza y permite al usuario entender el "porqué"de cada sugerencia.

2.3. Etapa 3: Síntesis de Contenido (Summarization)

Para maximizar la eficiencia del usuario, los artículos seleccionados son resumidos automáticamente. El sistema implementa una arquitectura de síntesis dual:

- **Summarizer Algorítmico (Baseline)**: Como base, se utiliza un extractor de resúmenes basado en el algoritmo **TextRank**, que identifica las oraciones más representativas de un texto basándose en un grafo de similitud. Es rápido y robusto.
- **Summarizer Neuronal (Avanzado)**: El sistema está preparado para integrar un modelo de Deep Learning abstracto, basado en una arquitectura **Pointer-Generator Network (PGN)**. Este tipo de modelo es capaz de entender el texto y generar un resumen nuevo, pudiendo copiar palabras textuales (puntero) o generar palabras nuevas del vocabulario (generador). Esto le permite manejar entidades y términos raros con mayor precisión que los modelos secuencia a secuencia tradicionales.

2.4. Etapa 4: Generación del Reporte Final

El producto final es un reporte profesional en formato PDF, generado mediante la librería `reportlab`. Este componente organiza toda la información curada y sintetizada en un documento coherente y fácil de leer, que incluye:

- Una portada con el título del reporte y la fecha de generación.
- Un resumen de los intereses del usuario o la consulta que originó el reporte.
- Una lista de los artículos recomendados, cada uno con su título, sección, fecha, el resumen generado y un enlace a la fuente original.

3. Componentes de Software y Flujo Interno

Esta sección detalla los componentes concretos que implementan el pipeline descrito anteriormente y cómo se comunican entre sí dentro del proyecto `Report_Generator`.

3.1. Capa de Ingesta y Preprocesamiento

En la primera etapa, el sistema construye y mantiene el corpus local de artículos a partir del canal de Telegram de Telesur:

- **Extracción desde Telegram**: El módulo `telegram/extract_data_tg.py` define la clase **ScraperT**, responsable de monitorear el canal de Telegram, descargar los mensajes y extraer las URLs de las noticias. Estos datos se almacenan como archivos JSON en directorios del tipo `Data/Data_articles<n>`, que constituyen el "cuerpo crudo" del sistema.
- **Preprocesamiento lingüístico**: El módulo `src/nlp/preprocessing.py` implementa la clase **TextPreprocessor**, que limpia el texto eliminando ruido editorial (patrones como "LEA TAMBIÉN"), normaliza espacios, tokeniza, elimina stopwords y, cuando es posible, aplica lematización en español mediante `spaCy`. Este preprocesador se utiliza tanto para artículos como para entradas de usuario.
- **Anotación por expresiones regulares**: El módulo `src/nlp/regex_annotator.py` define un anotador basado en reglas (**RegexAnnotator**) que detecta categorías temáticas específicas del dominio (política latinoamericana, conflictos internacionales, economía, derechos humanos, etc.). Estas categorías se guardan en cada artículo y se reutilizan más adelante en el motor de recomendación.

3.2. Vectorización, Perfiles de Usuario y Motor de Matching

La segunda etapa se materializa en una serie de componentes ubicados en el paquete `src/recommendation`:

- **Vectorizador de noticias**: El módulo `src/recommendation/vectorizer.py` implementa la clase **NewsVectorizer**, que envuelve un modelo TF–IDF de `scikit-learn`. Este vectorizador convierte los textos limpios de los artículos en vectores numéricos de alta dimensionalidad, reutilizables en la API de producción.
- **Perfiles vectoriales de usuario**: Sobre **NewsVectorizer** se construye **UserProfileVectorizer**, que recibe el texto de perfil del usuario (o su consulta en lenguaje natural) y produce vectores consistentes con el espacio semántico del corpus. El módulo `src/recommendation/user_profile.py` incluye **UserProfileManager**, encargado de extraer entidades con `spaCy`, consolidar categorías y generar un perfil enriquecido y persistente en `Data/Data_users/users.json`.
- **Motor de matching**: El archivo `src/recommendation/matcher.py` define la clase **NewsMatcher** (un matcher simplificado v6) que calcula la relevancia de cada artículo respecto al perfil combinado del usuario. El score final integra varios factores: similitud semántica por coseno entre vectores TF–IDF, coincidencia de categorías, coincidencia de entidades (personas, países, organizaciones) y un componente de recencia temporal basado en decaimiento

exponencial. El mismo módulo implementa estrategias de deduplicación y diversidad para evitar recomendaciones redundantes.

- **Estrategia de búsqueda por tiempo**: El módulo `src/process/news_recomendation.py` implementa el pipeline operativo de recomendación a través de funciones como `process_user_input`, `combine_user_profile_with_input` y `find_relevant_articles_with_time_strategy`. Esta última recorre los directorios `Data_articles n` empezando por los más recientes, aplica **NewsMatcher** y se detiene cuando se cumple un criterio de calidad (suficientes artículos altamente coincidentes) o un límite de tiempo global, equilibrando cobertura y latencia.

3.3. Síntesis de Contenido y Generación de Reportes

La tercera y cuarta etapa integran los módulos de resumen automático y de construcción de reportes finales:

- **Summarizers**: El módulo `src/summarization/summarizer.py` provee dos resumidores principales. **TextRankSummarizer** implementa un resumen extractivo ligero basado en posición y longitud de oraciones, mientras que **ModelSummarizer** actúa como fachada sobre un modelo *Pointer-Generator Network* entrenado específicamente, cuando el fichero de pesos está disponible. Adicionalmente, **PersonalizedSummarizer** permite ponderar oraciones según las categorías de interés del usuario.
- **Formateo intermedio de reportes**: El módulo `src/process/report_formatter.py` sirve de intermediario entre el pipeline de recomendación y el generador de reportes. La función `generate_report_from_user_query` orquesta el proceso de extremo a extremo: invoca a `generate_report_recommendations`, aplica el resumen automático a los artículos seleccionados y transforma las salidas del matcher en una estructura uniforme con justificaciones explicitadas (categorías y entidades coincidentes, descomposición de scores, estadísticas de búsqueda).
- **Generador de reportes**: El módulo `src/recommendation/report_generator.py` define la clase **ReportGenerator**, que cumple dos funciones: *i)* construir un diccionario de reporte estructurado listo para consumo tanto por la API como por el frontend, y *ii)* generar el PDF final con `reportlab` (títulos, metadatos, secciones de artículos, resúmenes, entidades destacadas y enlaces clicables). El método `generate_pdf` refleja en el diseño del documento los mismos elementos que se exponen en la versión de texto plano, manteniendo consistencia entre formatos.

3.4. API Web y Flujo de Peticiones

El proyecto ofrece una API ligera para consumo interactivo desde un frontend web o cliente externo. El archivo `src/api/api.py` implementa un servicio **FastAPI** que expone los siguientes elementos clave:

- **Gestión de usuarios**: Endpoints para registro y autenticación que crean y almacenan perfiles vectoriales en `users.json`, manteniendo la lógica de construcción de perfiles en **UserProfileManager** y **UserProfileVectorizer**.
- **Sesiones y contexto de conversación**: Un pequeño mecanismo de sesión en archivos JSON que permite conservar el contexto de interacción del usuario (por ejemplo, consultas previas y reportes generados).
- **Generación de reportes bajo demanda**: El endpoint `/recommendations/generate-text-report` recibe el ID de usuario y una consulta en lenguaje natural; internamente utiliza los componentes de inicialización de vectorizadores y matcher, llama a `generate_report_from_user_query` y devuelve tanto una versión de texto plano como una versión estructurada lista para ser convertida en PDF.

- **Gestión de PDFs**: Endpoints dedicados a generar y descargar PDFs (`/reports/generate-pdf` y `/reports/download-pdf`), que encapsulan el uso de **ReportGenerator** y administran el directorio de salida de documentos.

En conjunto, estos componentes implementan de forma explícita el flujo descrito en la sección de Metodología: desde la captura de noticias y el modelado del perfil semántico del usuario, hasta el cálculo de relevancia, la síntesis del contenido y la entrega final de un reporte justificable y portable.

4. Aportes y Valor del Proyecto

El valor de Report Generator no radica en un único algoritmo, sino en la **integración sinérgica de múltiples técnicas de NLP en un flujo de trabajo automatizado de extremo a extremo**.

- **De la Información al Conocimiento**: El proyecto materializa el concepto de convertir datos (noticias en bruto) en conocimiento accionable (un reporte personalizado y resumido), resolviendo un problema real para cualquier persona que necesite mantenerse informada sobre temas específicos.
- **Inteligencia Artificial Pragmática y Explicable**: En lugar de una *caja negra*, el sistema ofrece transparencia en sus recomendaciones. Este enfoque de IA Explicable (XAI) es fundamental para aplicaciones donde la confianza y la comprensión del usuario son prioritarias.
- **Flexibilidad Arquitectónica**: La arquitectura modular, como la capacidad de intercambiar modelos de resumen, demuestra un diseño robusto y preparado para futuras mejoras. La elección de una API ligera y persistencia en archivos es una decisión estratégica que favorece la portabilidad y la simplicidad en el despliegue, haciéndolo ideal para pruebas de concepto, entornos académicos o sistemas embebidos.

5. Conclusión

Report Generator es una demostración exitosa de cómo las técnicas modernas de Procesamiento del Lenguaje Natural pueden ser orquestadas para crear herramientas de productividad intelectual de alto impacto. El sistema no solo filtra información, sino que la sintetiza y la contextualiza, entregando un producto final que ahorra tiempo y mejora la comprensión.

El proyecto establece una base sólida sobre la cual se pueden construir futuras mejoras, como la integración de múltiples fuentes de noticias, análisis de sentimiento más profundos o la personalización del estilo de los resúmenes. Como solución autocontenida y funcional, cumple con su objetivo de transformar el consumo de noticias de una actividad pasiva a un proceso de inteligencia activa y personalizada.