



UNIVERSIDAD
CATÓLICA DE
TEMUCO

DEPARTAMENTO DE
INGENIERÍA INFORMÁTICA
FACULTAD DE INGENIERÍA

**Content Condenser: Aplicación para síntesis de información y
generación de resúmenes con modelos IA**

por

Josué Omar Rivas Vergara

Trabajo de Título presentado a la
Facultad de Ingeniería de la Universidad Católica de Temuco
Para Optar al Título de Ingeniero Civil Informático.

- Temuco, Mayo 2025 -



COMISION EXAMEN DE TITULO

Este Examen de Título ha sido realizado en la Escuela de Ingeniería Informática:

Presidente Comisión:

.....
Prof. Alejandro Mellado Gatica
Magíster en Ingeniería en Informática
Ingeniero Informático
Jefe Carrera Ing. Civil Informática

Profesor Guía:

.....
Prof. Nombre del Prof. Guía de Trabajo de Título
Título Profesional
Título de Posgrado más Alto

Profesor Informante:

.....
Prof. Nombre del Informante de Trabajo de Título
Título Profesional
Título de Posgrado más Alto

Jefe Carrera de Ingeniería
Civil Informática:

.....
Sr. Alejandro Mellado Gatica
Magíster en Ingeniería en Informática
Ingeniero Informático
Jefe Carrera Ing. Civil Informática

Temuco, día de mes de año.



INFORME TRABAJO DE TÍTULO

TÍTULO : “TÍTULO DEL TRABAJO DE TÍTULO”

ALUMNO : NOMBRE DEL ESTUDIANTE

En mi calidad de Profesor Guía, mis apreciaciones del presente informe de Trabajo de Titulo, son las siguientes:

- Apreciación 1.
- Apreciación 2.
- Apreciación 3.
- Etc...

De acuerdo a estas consideraciones califico el desarrollo de este Trabajo de Título con **nota X,X (nota en letras)**.

Guía

Prof. Nombre del Prof.

Profesor Guía

Temuco, dia de mes de año.



INFORME TRABAJO DE TÍTULO

TITULO : “TÍTULO DEL TRABAJO DE TÍTULO”

ALUMNO : NOMBRE DEL ESTUDIANTE

En mi calidad de Profesor Informante, mis apreciaciones del presente informe de Trabajo de Titulo, son las siguientes:

- Apreciación 1.
- Apreciación 2.
- Apreciación 3.
- Etc...

De acuerdo a estas consideraciones califico el desarrollo de este Trabajo de Título con **nota X,X (nota en letras)**.

Informante

Prof. Nombre del Profesor

Profesor Informante

Temuco, día de mes de año.

AGRADECIMIENTOS

Agradecimientos. Se pide que sean lo más cortos posibles.
maquete

Quiero expresar mi más sincero agradecimiento a todas las personas que han hecho posible la realización de esta tesis.

En primer lugar, agradezco a mi director/a de tesis, el Dr./Dra. [Nombre], por su orientación, apoyo y paciencia durante todo el proceso. Su experiencia y consejos fueron fundamentales para el desarrollo de este trabajo.

Agradezco a mis compañeros de la [Nombre de la carrera o programa], quienes hicieron que este camino fuera más llevadero, especialmente a [Nombres de compañeros], por sus valiosas discusiones y su ayuda en los momentos difíciles.

También quiero reconocer a mi familia por su amor incondicional y apoyo constante. A mis padres, [Nombres], gracias por darme la oportunidad de perseguir mis sueños y por siempre creer en mí. A [otro familiar, si es necesario], gracias por su aliento y motivación.

A la [Nombre de la institución o laboratorio, si aplica], por brindarme los recursos necesarios y un ambiente propicio para llevar a cabo mi investigación.

Finalmente, agradezco a todos aquellos que, de alguna manera, contribuyeron a esta etapa de mi vida. Sin su ayuda y apoyo, este trabajo no habría sido posible.

Índice general

Índice general	I
Índice de figuras	III
Índice de cuadros	IV
Resumen	V
1 Introducción	1
1.1. Problema	1
1.2. Estado del Arte	2
1.2.0.1. Brechas Identificadas	4
1.3. Objetivos	4
1.3.1. Objetivo general	4
1.3.2. Objetivos específicos	4
2 Marco teórico	6
2.1. Contenido del marco teórico	6
2.1.1. Procesamiento de Lenguaje Natural (NLP)	6
2.1.2. API de Transcripción de YouTube	6
2.1.3. Algoritmo TextRank para Resumen de Textos	7
2.1.4. Segmentación del Texto en Fragmentos	8
3 Desarrollo	9
3.1. Metodología	9
3.1.1. Plan de Acción para el Logro de Objetivos	11
3.2. Implementación	17
3.2.1. Diagrama de Flujo	17

3.2.2. Diagrama de Secuencia	20
3.2.3. Diagrama de Estado	20
3.2.4. Diagrama de Clases	21
3.3. Algoritmos	22
3.3.1. Pseudocódigo de los algoritmos usados	22
3.3.2. ¿Cómo funciona el sintetizador ?	24
4 Resultados	26
4.1. Datos	26
4.2. Plataforma informática de trabajo y parámetros de ejecución	27
4.3. Resultados de salida de programas	27
5 Discusión y Análisis de Resultados	30
5.1. Costo de tokens	30
5.1.1. Costos de créditos en OpenAI	30
5.1.2. Estimación por consulta	30
5.1.2.1. Nemotron-70b-Instruct	30
5.1.2.2. Nemoguard-8b	31
5.1.2.3. Qwen-32b	31
5.1.3. Comparación de Modelos	31
5.2. Optimización del Algoritmo mediante el Uso de Librerías como TextRank	32
5.3. Uso de umbral en lugar de sentencias	33
5.3.1. Comparación entre Métodos	34
6 Conclusiones	36
Bibliografía	37
7 Anexos	39
Anexo C: Código fuente completo	41

Índice de figuras

2.1. Ejemplo de Representación de un Grafo de TextRank.	8
3.1. Diagrama de Flujo del Proceso de Generación de Resúmenes y Cuestionarios.	10
3.2. Diferencias entre pytube y transcrip api.	11
3.3. Funcionamiento SpaCy	12
3.4. Diferencias entre SpaCy y NLTK	13
3.5. Diferencias entre los modelos para generación de cuestionarios	15
3.6. Comparación de Frameworks para Desarrollo Rápido en Python	16
3.7. Diagrama de flujo del proceso de la aplicación	19
3.8. Diagrama de secuencia de la interacción entre componentes .	20
3.9. Diagrama de estado del sistema	21
3.10. Diagrama de clases del sistema	22
4.1. Salida del resumen	28
4.2. Salida del cuestionario generado	29
5.1. Comparación del costo por consulta entre distintos modelos. .	32
5.2. Distribución de similitudes entre oraciones.	34
7.1. Interfaz del resumidor de videos de YouTube	39
7.2. Interfaz del generador de cuestionarios	40

Índice de cuadros

3.1. Comparación de modelos de lenguaje	14
3.2. Ejemplo de cálculo de similitud en TextRank.	17
4.1. Tipos de datos utilizados	26
4.2. Especificaciones de hardware y software	27
4.3. Valores de parámetros para la ejecución del sistema	27
4.4. Datos de salida del programa	28

Resumen

En el ámbito educativo, especialmente en entornos virtuales como Moodle o plataformas de gestión del aprendizaje, el uso de videos como material de enseñanza ha crecido de forma exponencial. Sin embargo, los estudiantes a menudo enfrentan dificultades para revisar o sintetizar estos contenidos de forma eficiente, lo que puede afectar la comprensión y el aprendizaje autónomo. En este contexto, surge la necesidad de desarrollar herramientas que automaticen la transcripción y el resumen de clases grabadas, permitiendo una mejor organización del contenido y facilitando el acceso a información clave.

Pese a la disponibilidad de grabaciones de clases, no existe una herramienta integrada que permita generar resúmenes precisos y accesibles de los contenidos expuestos en los videos. Esto genera una sobrecarga cognitiva en los estudiantes, quienes deben revisar material extenso sin ayuda estructurada. Además, muchos sistemas actuales no ofrecen segmentación coherente ni adaptaciones al idioma español, lo que limita su aplicabilidad en contextos educativos hispanohablantes.

Se desarrolló un sistema automatizado que permite transcribir y resumir videos de YouTube, orientado específicamente al uso educativo. La herramienta extrae subtítulos en español usando la biblioteca YouTubeTranscriptApi y aplica el algoritmo TextRank junto con spaCy para seleccionar las frases más relevantes. Para mejorar la coherencia del resumen, se incorpora una segmentación textual basada en fragmentos de aproximadamente 200 palabras. Además, se implementó la generación asistida de resúmenes utilizando la API de NVIDIA con el modelo nvidia/llama-3.1-nemotron-70b-instruct, logrando resultados más detallados y estructurados.

El sistema alcanzó una alta precisión en la transcripción de videos y generó resúmenes coherentes, reduciendo el texto sin comprometer la información esencial. En pruebas realizadas con videos de distintas disciplinas académicas, la combinación de TextRank, segmentación por fragmentos y asistencia con modelos de lenguaje avanzados mejoró la calidad de los resúmenes frente a enfoques tradicionales. Los resúmenes generados ayudaron a los estudiantes

a identificar rápidamente los conceptos clave, apoyando procesos de repaso y estudio autónomo.

La integración de técnicas de procesamiento de lenguaje natural y modelos de generación de texto permite optimizar la extracción de información educativa desde contenido audiovisual. Esta solución representa un aporte significativo al aprendizaje autónomo, reduciendo la carga de revisión de material extenso y facilitando el acceso estructurado al conocimiento.

Como proyección futura, se plantea extender esta herramienta con capacidades de generación automática de cuestionarios basados en los resúmenes, facilitando la autoevaluación del aprendizaje. También se considera su integración dentro de plataformas educativas como Moodle, mediante la incorporación como un complemento accesible para estudiantes y docentes. Finalmente, se buscará evaluar el impacto pedagógico del sistema mediante estudios controlados en entornos reales de enseñanza.

Palabras Clave: transcripción automática, resumen de texto, procesamiento de lenguaje natural, TextRank, NVIDIA API, YouTube, educación.

Capítulo 1

Introducción

1.1. Problema

Dentro del ámbito educativo actual, los estudiantes enfrentan una sobrecarga de contenido multimedia (clases grabadas, videos explicativos, documentos, ppt explicativos, entre otros), proporcionados por los docentes. Esta situación se ve agravada por la limitación de tiempo que los alumnos tienen para revisar y asimilar el contenido, lo que puede repercutir negativamente en su rendimiento académico

La ausencia de una síntesis de contenido y/o resumen estructurado impide que los estudiantes puedan concentrarse en los puntos clave de las lecciones, dificultando su capacidad para identificar información relevante y esencial

Ante dicha problemática, surge la necesidad de desarrollar herramientas que automaticen el proceso de síntesis de información y proporcionen a los estudiantes resúmenes organizados y accesibles. Estas herramientas podrán facilitar la mejor comprensión del contenido y optimizar el tiempo de estudio contribuyendo así a una mejoría en el rendimiento académico en el ámbito educativo

1.2. Estado del Arte

La síntesis automatizada de información y la generación de cuestionarios interactivos han evolucionado significativamente en la última década, impulsadas por avances en procesamiento de lenguaje natural(PLN) y aprendizaje automático.

Desde sus inicios, la generación de resúmenes automáticos ha sido abordada mediante enfoques extractivos y abstrativos. En los primeros trabajos, la técnica de resumen extractivo fue pionera, seleccionando oraciones clave de los textos para representar su contenido esencial (Luhn, 1958). Posteriormente, se introdujeron enfoques más avanzados con modelos de aprendizaje automático para mejorar la coherencia y relevancia de los resúmenes generados (Radev et al., 2004).

En la última década, se ha explorado el uso de redes neuronales y modelos de lenguaje preentrenados para mejorar la calidad de los resúmenes. Por ejemplo, See et al. (2017) propusieron un modelo de atención con cobertura que redujo significativamente la repetición en los resúmenes generados. Sin embargo, su enfoque estaba limitado a textos escritos y no consideraba transcripciones de videos.

En el ámbito de la educación, diversas investigaciones han abordado el uso de herramientas de PLN para mejorar la experiencia de aprendizaje. Gupta y Lehal (2019) desarrollaron un sistema que genera resúmenes automáticos de materiales educativos, incluyendo libros y artículos científicos. Aunque este trabajo demostró mejoras en la accesibilidad de la información, no integró la generación de cuestionarios interactivos como complemento al resumen.

Por otro lado, la generación de cuestionarios automáticos ha sido abordada en trabajos recientes. Kurdi et al. (2020) realizaron una revisión sobre la generación automática de preguntas (GAP), destacando el uso de modelos basados en reglas y enfoques neuronales para la creación de evaluaciones formativas. No obstante, estos estudios se han centrado principalmente en textos escritos y no en transcripciones de videos.

En el contexto de Moodle y plataformas de aprendizaje en línea, Alonso et al. (2021) propusieron una herramienta integrada que permite la generación de cuestionarios a partir de documentos académicos. Aunque su sistema logró una mejora en la automatización de la evaluación,

no incluyó la capacidad de sintetizar información desde videos ni la interacción con contenidos multimedia.

Por ende la propuesta planteada mejora los trabajos previos al integrar un sistema que no solo resume documentos escritos y transcripciones de videos de YouTube, sino que también genera cuestionarios interactivos basados en los resúmenes. A diferencia de investigaciones anteriores, este enfoque permite una mayor adaptabilidad a diferentes formatos de contenido educativo, ofreciendo una solución más integral para el aprendizaje asistido por IA.

Además de los avances en generación de resúmenes y cuestionarios, la extracción automatizada de texto a partir de videos ha sido un área activa de investigación. Los enfoques tradicionales se basaban en el reconocimiento de voz (ASR, *Automatic Speech Recognition*) y el procesamiento de subtítulos incrustados. Por ejemplo, Arora y Lahiri (2020) propusieron un *pipeline* para extraer y resumir contenido educativo de videos mediante ASR, aunque su sistema no integraba generación de preguntas.

Un avance significativo llegó con modelos de PLN aplicados a transcripciones de videos. Liu et al. (2021) desarrollaron un marco para procesar textos extraídos de conferencias académicas, combinando ASR con técnicas de resumen abstractivo. Sin embargo, su enfoque no consideraba la diversidad de estilos en videos informales (e.g., YouTube).

Más recientemente, Wang et al. (2022) abordaron la extracción y síntesis de información desde videos multimodales (audio, texto visual y hablado), utilizando *transformers* para fusionar estas fuentes. No obstante, su trabajo se centró en la generación de resúmenes, dejando de lado aplicaciones educativas interactivas.

En el contexto específico de plataformas educativas, Chen et al. (2021) diseñaron un sistema que extrae transcripciones de videos en Moodle y genera apuntes estructurados. Pese a su utilidad, no exploraron la generación de cuestionarios ni la adaptabilidad a distintos dominios temáticos.

1.2.0.1. Brechas Identificadas

Estos trabajos evidencian que, aunque existen soluciones para extraer y procesar texto de videos, persisten **brechas clave**:

- **Integración con generación de cuestionarios:** La mayoría se limita a resúmenes o apuntes.
- **Adaptabilidad a contenido informal:** Los modelos suelen entrenarse con videos académicos (e.g., conferencias), no con el estilo dinámico de plataformas como YouTube.
- **Interactividad:** Pocos sistemas cierran el ciclo aprendizaje-evaluación automática.

1.3. Objetivos

1.3.1. Objetivo general

- Desarrollar una herramienta basada en inteligencia artificial que sintetice información a partir de links de YouTube y documentos en formato PDF, DOCX y PPT, permitiendo la generación de resúmenes automáticos y cuestionarios interactivos para el apoyo del aprendizaje.

1.3.2. Objetivos específicos

1. Implementar un sistema de transcripción automática, el cual a partir de un link de un video de YouTube extraiga el texto en un formato estructurado.
2. Diseñar un modelo de procesamiento de lenguaje natural que genere resúmenes automáticos a partir de textos extraídos de videos y documentos.
3. Desarrollar un módulo de generación automática de cuestionarios a partir de los resúmenes de los videos, utilizando técnicas de procesamiento de lenguaje natural y generación de preguntas con el apoyo de la api de gpt.

4. Implementar el algoritmo en un sitio web de prueba a través del uso de un framework.

Capítulo 2

Marco teórico

2.1. Contenido del marco teórico

El presente trabajo de título se enfoca en la transcripción y resumen automático de videos educativos utilizando herramientas de procesamiento de lenguaje natural (NLP). A continuación, se presentan los fundamentos teóricos necesarios para comprender el desarrollo del sistema propuesto.

2.1.1. Procesamiento de Lenguaje Natural (NLP)

El procesamiento de lenguaje natural (NLP) es un campo de la inteligencia artificial que permite a las computadoras entender, interpretar y generar lenguaje humano. Se basa en diversas técnicas como la tokenización, el etiquetado de partes de la oración y la extracción de información relevante. En el contexto de este trabajo, el NLP se emplea para procesar las transcripciones de videos y generar resúmenes automáticos.

2.1.2. API de Transcripción de YouTube

YouTube proporciona una API que permite acceder a las transcripciones de los videos que disponen de subtítulos automáticos o manuales. Esta API es clave en el sistema desarrollado, ya que permite obtener el texto de los videos sin necesidad de procesar el audio mediante reconocimiento

de voz. Una vez obtenida la transcripción, el texto es procesado y segmentado para su posterior análisis.

2.1.3. Algoritmo TextRank para Resumen de Textos

El algoritmo TextRank es una técnica basada en grafos utilizada para la extracción de resúmenes automáticos. Funciona asignando pesos a las oraciones de un texto según su relevancia dentro del documento. La representación de un texto mediante un grafo dirigido se define como:

$$G = (V, E) \quad (2.0)$$

donde V representa el conjunto de oraciones y E las conexiones ponderadas entre ellas. El peso de cada conexión se calcula mediante la similitud coseno entre las oraciones:

$$S(i, j) = \frac{\sum_k W_{ik} W_{jk}}{\sqrt{\sum_k W_{ik}^2} \sqrt{\sum_k W_{jk}^2}} \quad (2.0)$$

donde W_{ik} y W_{jk} representan la frecuencia de los términos en cada oración. Finalmente, se aplica el algoritmo de PageRank para determinar la importancia relativa de cada oración en el conjunto del texto.

El cálculo de la relevancia de cada oración $S(V_i)$ se obtiene iterativamente utilizando la siguiente ecuación:

$$S(V_i) = (1 - d) + d \sum_{V_j \in In(V_i)} \frac{S(V_j)}{|Out(V_j)|} \quad (2.0)$$

donde:

- $S(V_i)$ representa la puntuación de la oración V_i .
- d es el factor de amortiguamiento, con un valor común de 0.85.
- $In(V_i)$ es el conjunto de nodos que enlazan a V_i .
- $Out(V_j)$ es el número de enlaces salientes de V_j .

Este método permite identificar las oraciones más representativas del texto, generando un resumen basado en aquellas con mayor puntuación. TextRank se basa en la idea de que una oración es más importante si está conectada a otras oraciones importantes dentro del documento.

La Figura 2.1 representa un grafo generado por el algoritmo TextRank, donde cada nodo es un fragmento de texto ('chunk') y las aristas representan la similitud entre ellos. El grosor de las conexiones indica la fuerza de la relación, lo que permite identificar los fragmentos más relevantes para la generación del resumen.

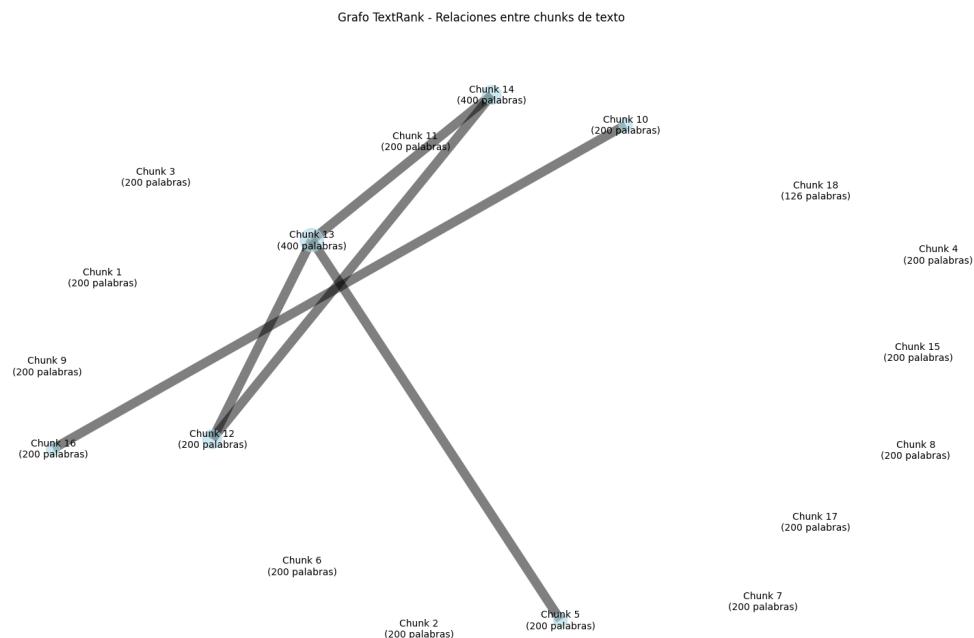


Figura 2.1: Ejemplo de Representación de un Grafo de TextRank.

2.1.4. Segmentación del Texto en Fragmentos

Dado que las transcripciones pueden ser extensas, se requiere dividir el texto en fragmentos manejables antes de aplicar el resumen. La división puede realizarse en función del número de palabras en cada segmento, asegurando que cada fragmento mantenga coherencia semántica y contextual.

Capítulo 3

Desarrollo

3.1. Metodología

Lo primero que se realizó para confeccionar la implementación es definir un flujo de características que va a tener la implementación final en el sistema web en donde en donde se gestionará la transcripción, el resumen y la generación de cuestionarios. Además del modelo de representación basado en grafos, el sistema sigue un flujo de procesamiento estructurado para la generación de resúmenes y cuestionarios a partir de distintos tipos de entrada. La Figura 3.1 ilustra el proceso general, desde la carga del contenido hasta la interacción final con el usuario.

Proceso de Generación de Resúmenes y
Cuestionarios

Figura 3.1: Diagrama de Flujo del Proceso de Generación de Resúmenes y Cuestionarios.

El proceso comienza con la recepción de una entrada, que puede ser un archivo de texto (PDF, DOCX, PPT) o un enlace a un video de YouTube. A partir de esta entrada, se extrae el texto o se descarga la transcripción en caso de videos. Luego, el contenido pasa por una fase de preprocesamiento, donde se normaliza el texto eliminando ruido y asegurando su compatibilidad con las siguientes etapas.

Posteriormente, se generan los resúmenes y los cuestionarios de manera independiente. Los resúmenes pueden visualizarse directamente, mientras que los cuestionarios permiten una evaluación interactiva del contenido, facilitando el proceso de estudio y comprensión del material.

3.1.1. Plan de Acción para el Logro de Objetivos

Para cumplir el objetivo 1, se estudiaron y probaron distintas bibliotecas de python que fueran capaces de transcribir audio a texto. Para ello se hicieron pruebas con las bibliotecas 3.2, como se observa en la tabla comparativa, luego de extensas pruebas en donde se consideraron los cambios futuros y la escalabilidad se optó por usar YouTube transcripción API, ya que esta última cuenta con una mejor precisión en cuanto al texto obtenido de los videos, así como la capacidad para modificar los parámetros de lenguaje (idioma latinoamericano específico en el cual obtener la transcripción)

Diferencias	pytube	Transcrip API
Tipo de herramienta	Biblioteca Python para descarga de subtítulos	API para transcripción de audio
Función Principal	Descarga videos de YouTube	Convierte audio de videos a texto
Requisitos	pip install pytube	pip install youtube-transcript-api
Proceso de trabajo	- proporcionar URL del video - Seleccionar formato - Seleccionar tipo de operación(descarga de audio o subtítulos)	- Proporcionar ID del video - Parsear tipo de acción requerida
Limitaciones	- No maneja videos privados - Si el video no contiene subtítulos no se obtiene la transcripción - Solo es compatible con versiones antigüas de python	- Requiere conexión estable - No reconoce ciertas palabras de ascendencia latina - Censura de palabras o modismos
Ejemplo de código	from pytube import YouTube # Descargar un video de YouTube video = YouTube("https://www.youtube.com/watch?v=YOUR_VIDEO_ID") stream = video.streams.filter(res="720p").first() stream.download(output_path="ruta/donde/guardar")	from youtube_transcript_api import YouTubeTranscriptApi # Obtener la transcripción de un video de YouTube transcript = YouTubeTranscriptApi.get_transcript("YOUR_VIDEO_ID") for entry in transcript: print(f'{entry["start"]}: {entry["text"]}')

Figura 3.2: Diferencias entre pytube y transcrip api.

Para cumplir el objetivo 2 se estudió el funcionamiento de librerías que trabajen con lenguaje natural o NLP. Para ello librerías como SpaCy y natural lenguaje tol kit(NLTK) fueron las principales opciones con las

que se trabajaron, cada una de estas librerías tenía una función única siendo finalmente Spacy la opción seleccionada principalmente porque a diferencia de NLTK esta cuenta con modelos pre entrenados capaces de reconocer palabras únicas dado diferentes variaciones del lenguaje (español y modismos latinos). La librería de Spacy para trabajar con textos funciona de la siguiente manera 3.3 las principales diferencias 3.4 entre ambas librerías radican en el conocimiento base que se requiere para trabajar con ellas siendo NLTK la más compleja ya que ademas de ser una librería mas robusta en términos de procesar texto es decir demora mas ya que es mas precisa en reconocer palabras, para este caso no resultó ser compatible con textrank, la cual es la librería que se encarga de generar los resúmenes automatizados

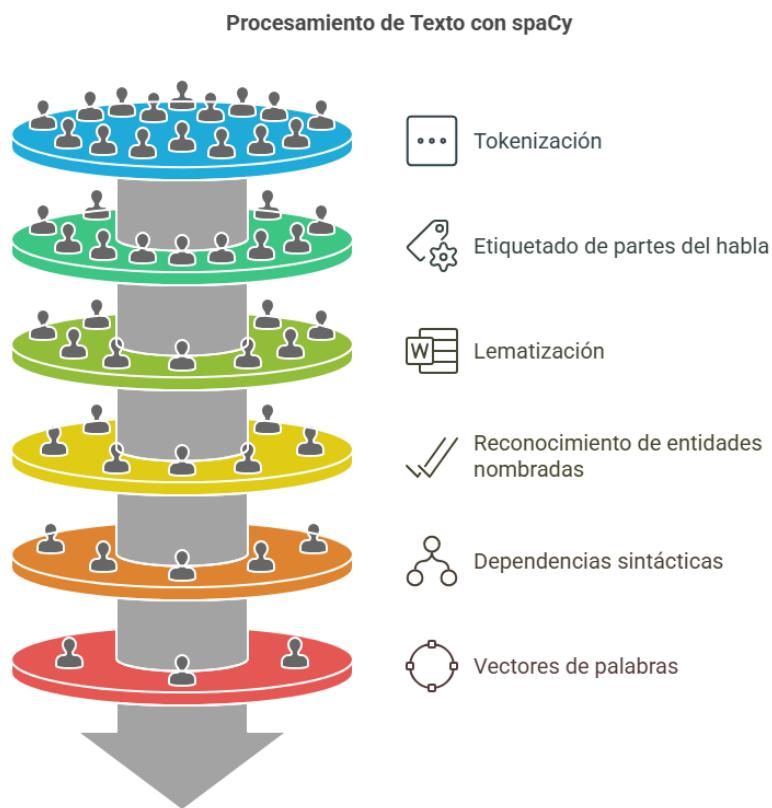


Figura 3.3: Funcionamiento SpaCy

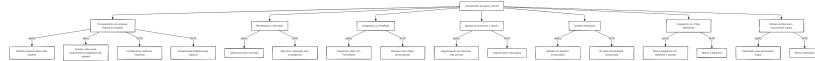


Figura 3.4: Diferencias entre SpaCy y NLTK

Para cumplir el objetivo 3 se analizó el costo computacional y los requisitos a nivel de hardware para poder ejecutar tres distintos modelos los cuales están enfocados en generación de cuestionarios. Dichas pruebas incluyeron consultas directas a cada uno de los modelos donde a partir de un texto extraído de un video, se les hacía la siguiente consulta:

```
1 prompt = '''Genera un cuestionario de 5 preguntas
2 en formato JSON. Cada pregunta debe tener:
3     - 4 opciones (a, b, c, d)
4     - 1 respuesta correcta
5     - Explicacion breve de la respuesta
6 Ejemplo de formato:
7 {
8     "preguntas": [
9         {
10             "pregunta": "texto",
11             "opciones": {"a": "op1", "b": "op2",
12                         "c": "op3", "d": "op4"},
13             "respuesta": "a",
14             "explicacion": "texto"
15         }
16     ]
17 }, '''
```

De esta consulta se obtuvieron los siguientes resultados luego del análisis de las respuestas 3.1, además de el gráfico que nos muestra de manera mas visual lo descrito en el cuadro 3.5

Característica	nemotron-70b	nemoguard-8b	Qwen-32B
Tamaño del modelo	70 mil millones de parámetros	8 mil millones de parámetros	32 mil millones de parámetros
Propósito principal	Generación de respuestas útiles y precisas	Moderación de contenido y clasificación de seguridad	Generación de texto generalista y comprensión del lenguaje
Ventajas	<ul style="list-style-type: none"> - Alto rendimiento en tareas de comprensión y generación de texto - Mejorado mediante Aprendizaje por Refuerzo con Retroalimentación Humana (RLHF) - Excelente desempeño en benchmarks de alineación automática 	<ul style="list-style-type: none"> - Especializado en la detección y clasificación de contenido inseguro - Entrenado en el conjunto de datos Aegis 2.0 para moderar interacciones entre humanos y modelos de lenguaje 	<ul style="list-style-type: none"> - Buen desempeño en generación de texto generalista - Optimizado para múltiples tareas en comprensión del lenguaje natural - Desarrollado por Alibaba Cloud con capacidades avanzadas
Desventajas	<ul style="list-style-type: none"> - Mayor costo operativo debido a su tamaño 	<ul style="list-style-type: none"> - Menor capacidad para tareas generales de generación de texto debido a su enfoque en seguridad 	<ul style="list-style-type: none"> - Requiere más recursos computacionales en comparación con modelos más pequeños - No especializado en tareas educativas como generación de cuestionarios

Cuadro 3.1: Comparación de modelos de lenguaje

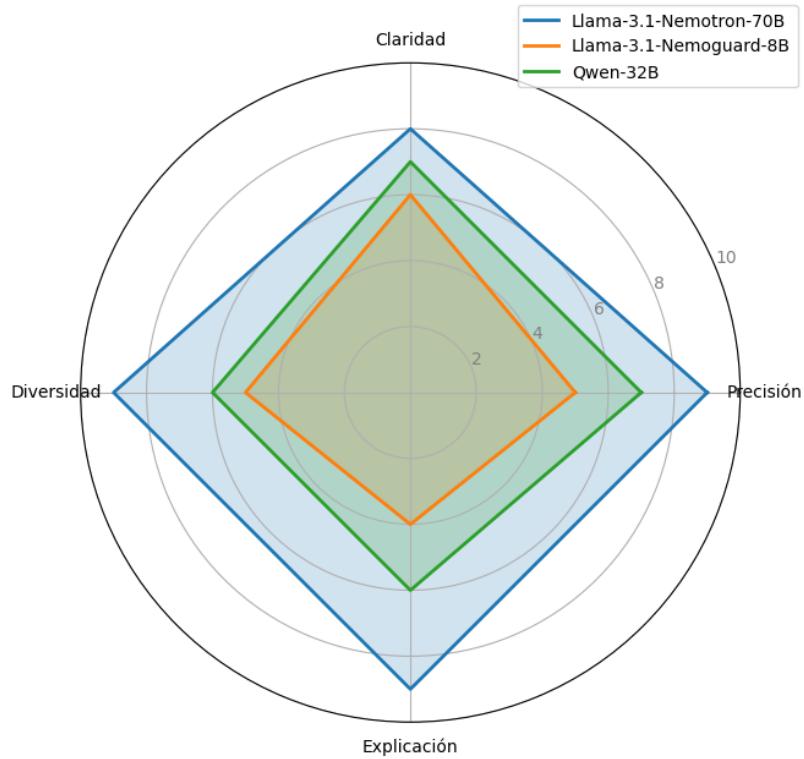


Figura 3.5: Diferencias entre los modelos para generación de cuestionarios

Para cumplir con el objetivo 4 se investigó el uso de distintos frameworks de python, los cuales permitirían montar el algoritmo en un sitio web para hacer pruebas en tiempo real del uso e implementación del algoritmo en un ambiente que fuera amigable para el usuario común, para ello se buscaron 4 puntos que determinarían donde es mas factible desplegar la herramienta dichos puntos son:

1. Facilidad de uso
2. Flexibilidad
3. Rapidez de desarrollo
4. Visualización de datos

dichos puntos fueron puestos a prueba y los resultados fueron 3.6, en donde **Streamlit (verde)** sobresale en facilidad de uso y rapidez de desarrollo, lo que lo hace ideal para montar una web rápida con Python. Además, su integración con gráficos y visualización de datos es superior.

Flask (azul) es más flexible y ligero, pero requiere más configuración manual, lo que puede ralentizar el desarrollo.

Django (rojo) es potente y escalable, pero es más complejo y requiere más tiempo de configuración inicial.

Dado esto se optó por montar la herramienta en streamlit dado sus ventajas en facilidad y rapidez en comparación con los otros dos frameworks

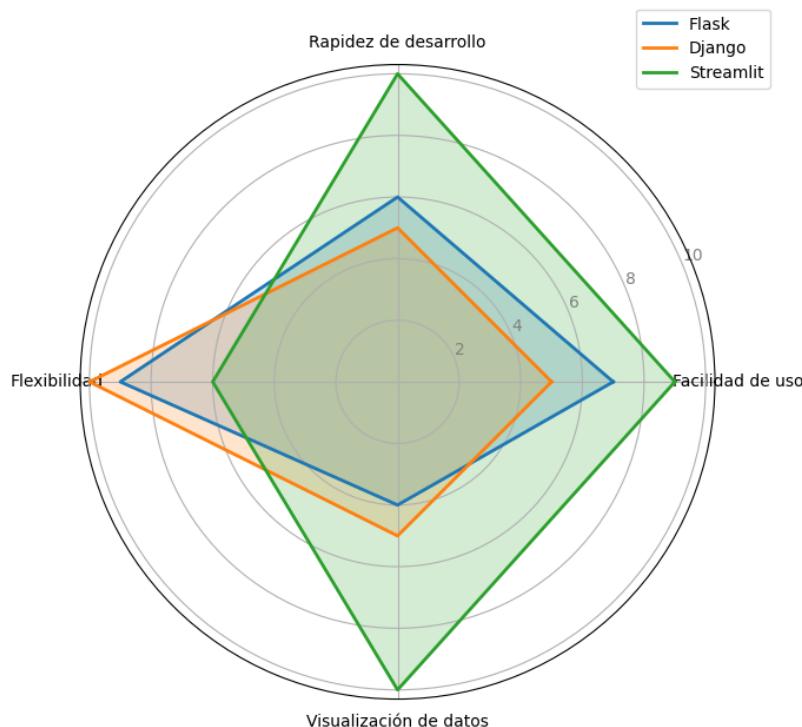


Figura 3.6: Comparación de Frameworks para Desarrollo Rápido en Python

En el Cuadro 3.2, se presentan ejemplos de fragmentos de texto extraídos de un documento y sus puntuaciones de similitud, obtenidas mediante

la métrica de similitud coseno.

Cuadro 3.2: Ejemplo de cálculo de similitud en TextRank.

Chunk	Contenido	Similitud
1	TextRank es un algoritmo de resumen basado en grafos.	1.00
2	Los nodos en TextRank representan fragmentos de texto.	0.85
3	Las aristas indican la similitud semántica entre fragmentos.	0.78
4	Se pueden usar diferentes métricas para evaluar la similitud.	0.60
5	El resumen final se construye con los fragmentos más relevantes.	0.92

3.2. Implementación

En este apartado se presentan los diagramas necesarios para implementar el algoritmo en un sistema web, en donde se incluyen aquellos esquemas que ilustran el flujo de trabajo, la interacción entre componentes y la estructura del sistema. Estos diagramas han sido diseñados para facilitar la comprensión del proceso y la interacción entre las diferentes partes del sistema.

Además, para complementar la implementación descrita, se incluyen en los Anexos A y B capturas completas de la interfaz desarrollada. En el 7.1 se presenta la vista del módulo de generación de cuestionarios a partir de videos de YouTube, mientras que en el 7.2 se muestra la interfaz correspondiente al generador de resúmenes y la visualización de la transcripción completa. Estas imágenes permiten observar el funcionamiento del sistema desde la perspectiva del usuario final y refuerzan los resultados obtenidos a lo largo del desarrollo.

3.2.1. Diagrama de Flujo

El diagrama de flujo (Figura 3.7) describe el proceso general de la herramienta, desde el inicio hasta la finalización del proceso.

Este diagrama incluye las diferentes opciones que el usuario puede seleccionar, como la extracción de transcripciones de YouTube, la generación de cuestionarios y la creación de resúmenes automáticos. Además, se detallan los pasos en caso de errores y cómo se manejan las excepciones.

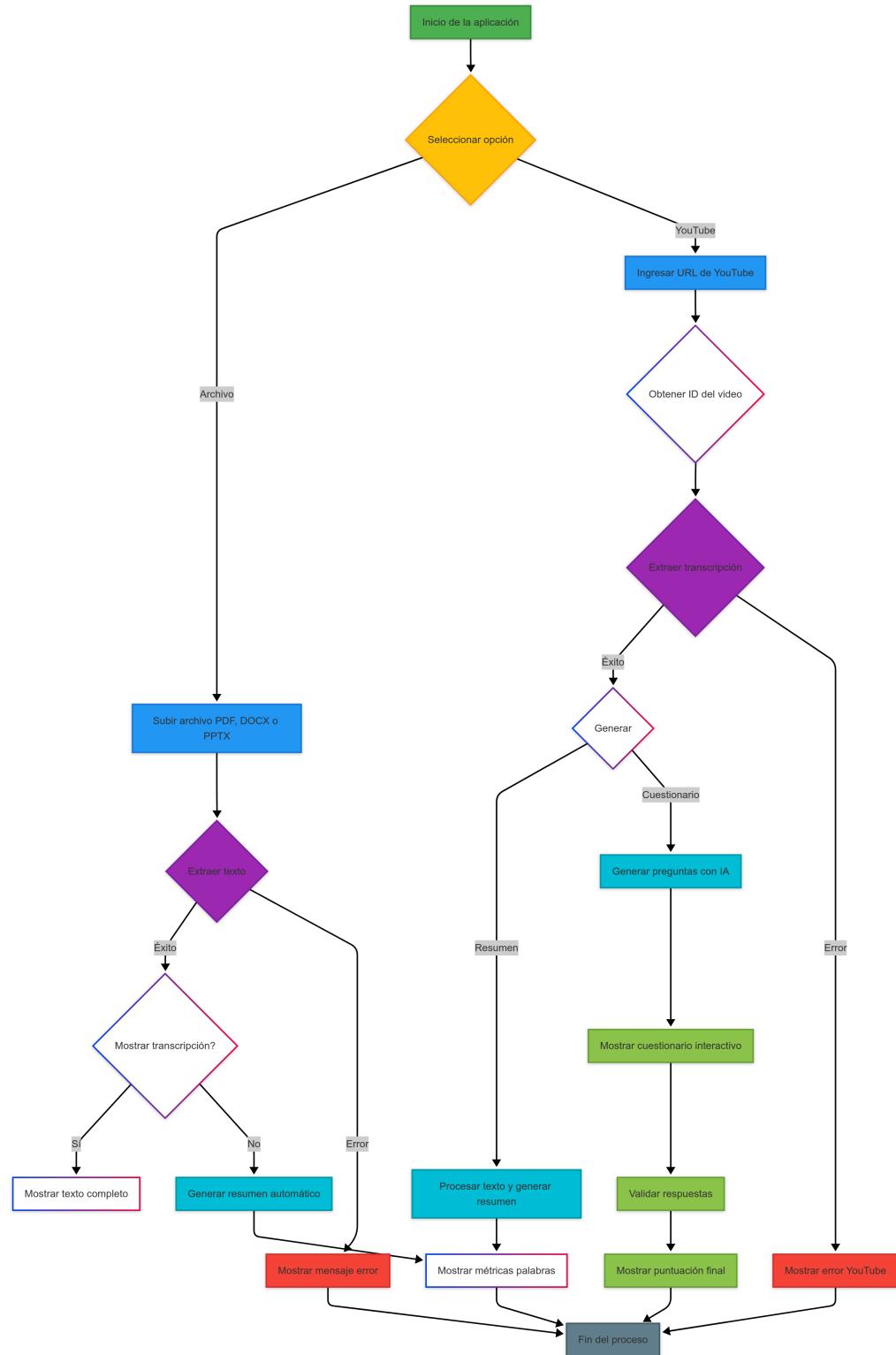


Figura 3.7: Diagrama de flujo del proceso de la aplicación

3.2.2. Diagrama de Secuencia

El diagrama de secuencia (Figura 3.8) muestra la interacción entre los diferentes componentes del sistema durante la ejecución de las funciones principales. Este diagrama detalla cómo se procesan las solicitudes, cómo se extraen los datos y cómo se generan los resultados finales, como cuestionarios y resúmenes.

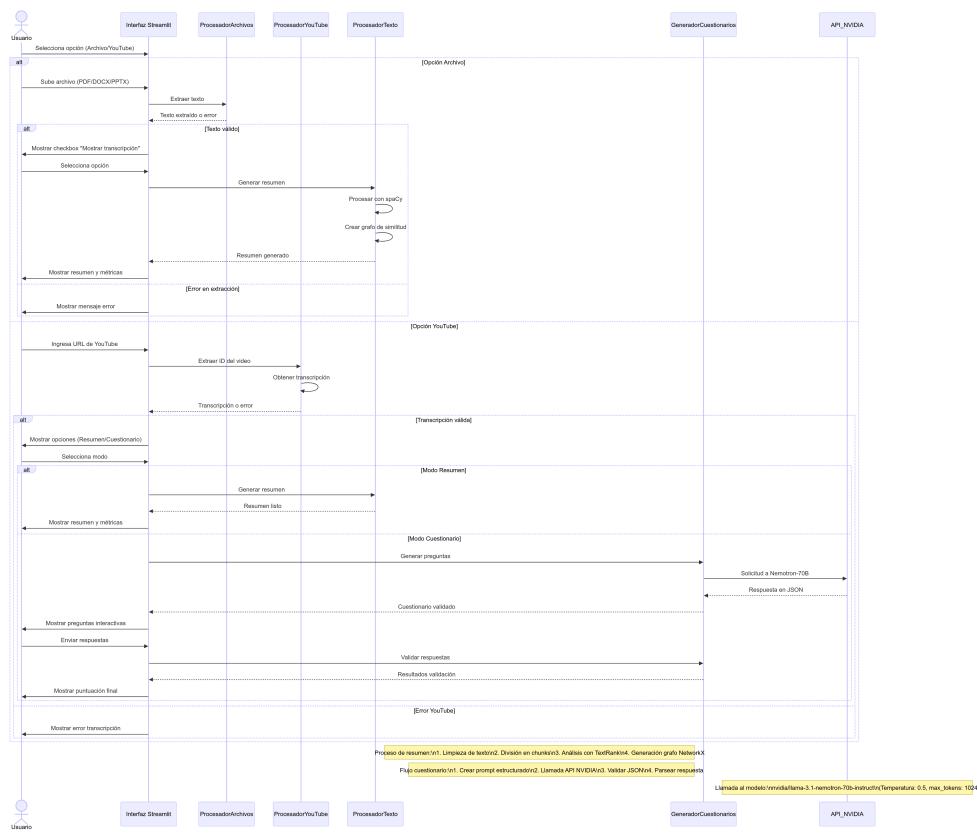


Figura 3.8: Diagrama de secuencia de la interacción entre componentes

3.2.3. Diagrama de Estado

El diagrama de estado (Figura 3.9) ilustra los diferentes estados por los que pasa el sistema durante la ejecución de las funciones relacionadas con la extracción de transcripciones de YouTube y la generación de

contenido. Este diagrama ayuda a comprender cómo el sistema maneja las transiciones entre estados y cómo se gestionan los errores.

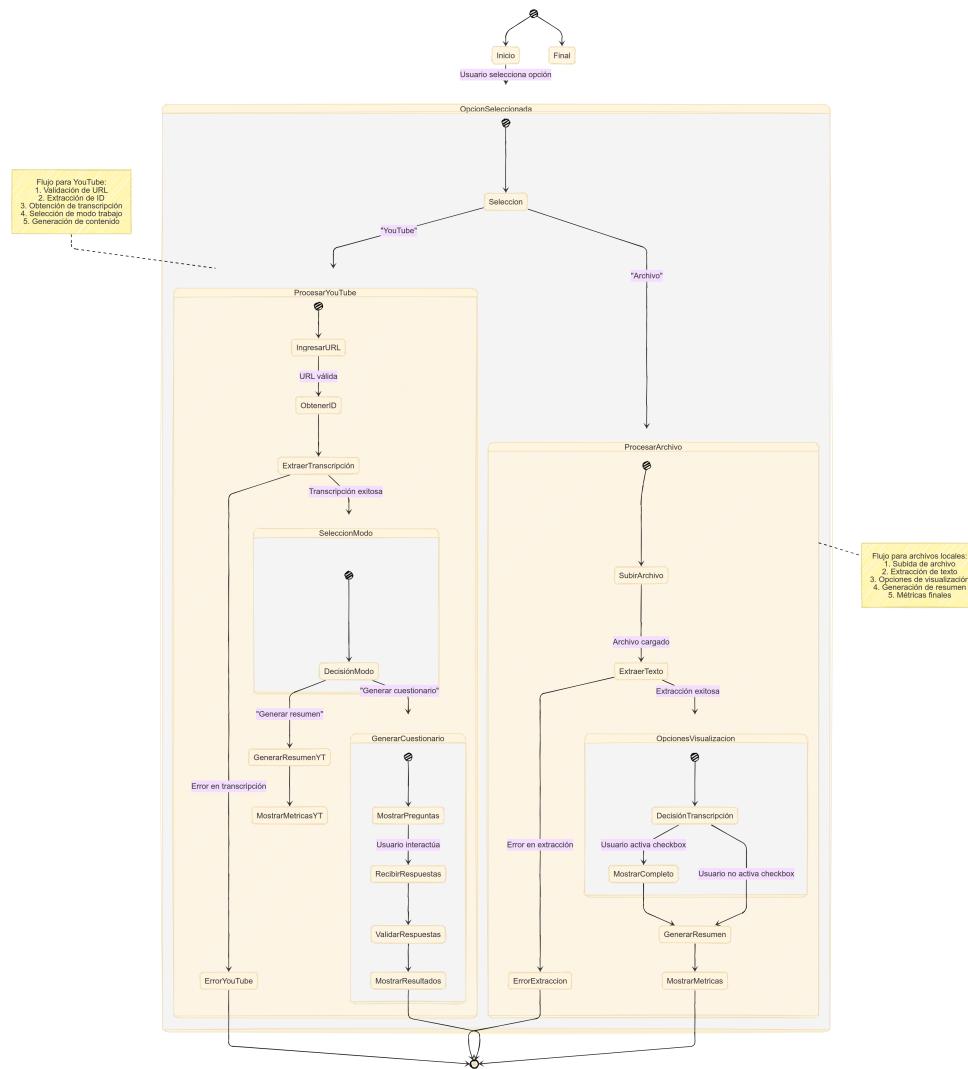


Figura 3.9: Diagrama de estado del sistema

3.2.4. Diagrama de Clases

El diagrama de clases (Figura 3.10) representa la estructura de las clases y la relación entre ellas. Este diagrama es fundamental para entender cómo

se organiza el código y cómo interactúan las diferentes partes del sistema. Además, se detallan las funciones principales y cómo se relacionan con los paquetes externos utilizados.

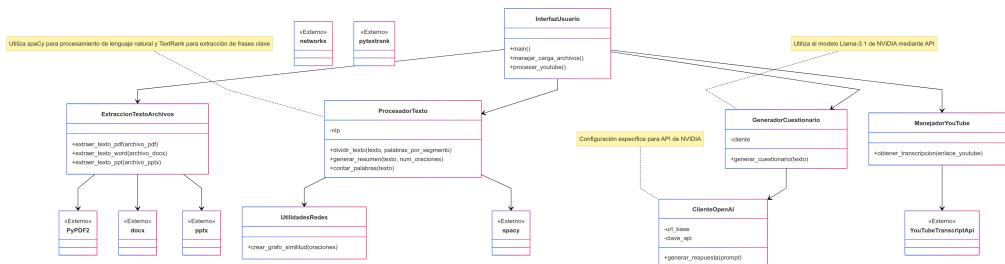


Figura 3.10: Diagrama de clases del sistema

3.3. Algoritmos

El proceso de generación de resúmenes se basa en la transcripción de videos de YouTube o en la extracción de texto desde documentos. Para ello, se aplica un modelo de segmentación de texto seguido de un algoritmo de resumen basado en extracción de frases clave mediante TF-IDF y TextRank.

El algoritmo 2.1.3 toma como entrada un texto transcrit o extraído y devuelve un resumen optimizado de acuerdo con el número de frases más representativas.

3.3.1. Pseudocódigo de los algoritmos usados

Algoritmo 1 GENERACIÓN DE RESUMEN POR EXTRACCIÓN CON TF-IDF Y TEXTRANK

```

1: Entrada: Texto transcrit o extraído  $T$ , número de frases deseadas en el resumen  $N$ 
2: Salida: Resumen  $S$ 
3: procedure GENERARRESUMEN( $T, N$ )
4:    $Frases \leftarrow \text{SEGMENTARTEXTO}(T)$ 
5:    $MatrizTFIDF \leftarrow \text{CALCULARTFIDF}(Frases)$ 
     ▷ Construcción de la matriz de similitud del coseno
6:   for  $i \leftarrow 1$  to  $|Frases|$  do
7:     for  $j \leftarrow i + 1$  to  $|Frases|$  do
8:        $Similitud[i][j] \leftarrow \text{SIMILITUDCOSENO}(MatrizTFIDF[i], MatrizTFIDF[j])$ 
9:        $Similitud[j][i] \leftarrow Similitud[i][j]$ 
10:    end for
11:   end for
12:    $GrafoSimilitud \leftarrow \text{CONSTRUIRGRAFO}(Similitud)$ 
13:    $Puntajes \leftarrow \text{APLICARTEXTRANK}(GrafoSimilitud)$ 
14:    $S \leftarrow \text{SELECCIONARMEJORESFRASES}(Frases, Puntajes, N)$ 
15:   return  $S$ 
16: end procedure

```

Algoritmo 2 CÁLCULO DE SIMILITUD DEL COSENO

```

1: procedure SIMILITUDCOSENO( $A, B$ )
2:    $\text{Numerador} \leftarrow \sum_{i=1}^m A_i B_i$ 
3:    $\text{Denominador} \leftarrow \sqrt{\sum_{i=1}^m A_i^2} \times \sqrt{\sum_{i=1}^m B_i^2}$ 
4:   if  $\text{Denominador} = 0$  then
5:     return 0
6:   else
7:     return  $\text{Numerador} / \text{Denominador}$ 
8:   end if
9: end procedure

```

3.3.2. ¿Cómo funciona el sintetizador ?

El sistema crea resúmenes en 4 etapas clave:

1. **Obtención del texto:** Extrae las palabras de documentos (PDF, Word, PowerPoint) o videos de YouTube.

2. **Preparación del contenido:**

- Divide el texto en frases individuales.
- Identifica las palabras importantes usando TF-IDF (fórmula que valora palabras poco comunes).
- Convierte cada frase a números (vectores) según sus palabras clave.

3. **Comparación de frases:**

- Calcula qué tan parecidas son las frases entre sí usando:

$$\text{Similitud} = \frac{\text{Puntos en común}}{\text{Largo frase A} \times \text{Largo frase B}} \quad (3.0)$$

- Crea un "mapa de relaciones" donde las frases más similares se conectan.

4. **Selección del resumen:**

- Usa el algoritmo TextRank (similar al de Google para páginas web) para encontrar:
 - Frases que aparecen en muchas conexiones.
 - Frases con conexiones fuertes.
- Elige las frases más "importantes" según el umbral definido, ordenadas según aparecen en el texto original.

Comparación de frases:

- Calcula qué tan parecidas son las frases entre sí usando:

$$\text{Similitud} = \frac{\text{Puntos en común}}{\text{Largo frase A} \times \text{Largo frase B}} \quad (3.0)$$

- Crea un "mapa de relaciones" donde las frases más similares se conectan

Selección del resumen:

- Usa el algoritmo TextRank (como el de Google para páginas web) para encontrar:
 - Frases que aparecen en muchas conexiones
 - Frases con conexiones fuertes
- Elige las frases más importantes "si estas tiene una puntuación sobre el umbral"
- Las ordena según aparecen en el texto original

En simple: El sistema funciona como un detective que: 1) Recolecta información, 2) Marca pistas clave, 3) Busca conexiones entre ideas, y 4) Arma la historia principal con las partes más conectadas.

Capítulo 4

Resultados

4.1. Datos

En este trabajo de título, se utilizan datos provenientes de enlaces de videos de YouTube que contienen clases grabadas por docentes. Actualmente, la recopilación de estos enlaces se realiza de manera manual, insertándolos en la plataforma desarrollada para su procesamiento. Además, se permite la carga de documentos en formatos PDF, DOCX y PPTX para su posterior análisis y síntesis.

Los datos extraídos incluyen la transcripción de los videos, la cual es procesada mediante algoritmos de procesamiento de lenguaje natural para generar resúmenes y, en una etapa futura, cuestionarios automatizados basados en el contenido analizado.

A continuación, se describe la estructura de los datos utilizados en el sistema:

Cuadro 4.1: Tipos de datos utilizados

Tipo de Dato	Descripción
Enlaces de YouTube	URLs de videos de clases subidos por docentes.
Transcripciones de video	Texto generado a partir del contenido hablado en los videos.
Archivos de texto	Documentos en formato PDF, DOCX y PPTX subidos por los usuarios.
Resúmenes	Síntesis del contenido extraído de los videos y documentos.
Cuestionarios	Generación automatizada de preguntas basadas en los resúmenes.

Este esquema de datos permite procesar y estructurar la información de manera eficiente, facilitando la generación de material de apoyo para los estudiantes.

4.2. Plataforma informática de trabajo y parámetros de ejecución

Para la ejecución del sistema de transcripción y resumen de videos, se utilizó un entorno de trabajo compuesto por hardware y software específico, descrito en la Tabla 4.2. La plataforma permite procesar videos de YouTube y documentos en distintos formatos (PDF, DOCX y PPTX) para generar transcripciones, resúmenes y, en una etapa futura, cuestionarios automatizados.

Cuadro 4.2: Especificaciones de hardware y software

Hardware	Especificación
Especificaciones de CPU y Memoria	
Modelo de CPU	Intel Core i7-8750H
Cantidad de núcleos	6
Cantidad de hilos	8
Frecuencia base	2.3 GHz
Memoria RAM	16GB DDR4
Especificaciones de GPU (si aplica)	
Modelo de GPU	NVIDIA RTX 1050
Memoria de GPU	4GB GDDR4
Software	Especificación
Sistema Operativo	Windows 11 / Ubuntu 22.04
Lenguaje de Programación	Python 3.10
Entorno de Desarrollo	Jupyter Notebook / Google Colab
Librerías utilizadas	Streamlit, PyPDF2, docx, pptx, LangChain, NLTK, Scikit-learn, NumPy, NetworkX, pytextrank, Spacy, networkx, YouTubeTranscriptApi, Openai

El algoritmo se ejecuta en la plataforma descrita en la Tabla 4.2. A continuación, en la Tabla 4.3, se presentan los parámetros de configuración utilizados en el procesamiento de transcripciones y resúmenes.

Cuadro 4.3: Valores de parámetros para la ejecución del sistema

Configuración	Valores
Duración máxima de video procesado	30 minutos
Fragmentación de texto	500 palabras
Método de resumen	TextRank / LangChain
Formato de salida	TXT, JSON
Número máximo de preguntas en cuestionario	10
Modelo de lenguaje utilizado	llama-3.1-nemotron-70b-instruct

4.3. Resultados de salida de programas

El sistema desarrollado genera diferentes tipos de datos de salida, los cuales incluyen transcripciones de videos, resúmenes de contenido y, de manera opcional, cuestionarios basados en los resúmenes. Estos resultados permiten a los usuarios obtener una síntesis del material procesado y evaluar su comprensión mediante preguntas generadas automáticamente.

En la Tabla 4.4 se detallan los tipos de salida generados por el sistema. Además, en la Figura 4.1 se muestra un ejemplo de transcripción y resumen generados a partir de un

video, mientras que la Figura 4.2 presenta un cuestionario autogenerado basado en el resumen.

Cuadro 4.4: Datos de salida del programa

Tipo de Salida	Descripción
Transcripción de video	Texto extraído del audio de los videos procesados.
Resumen del contenido	Síntesis generada a partir de la transcripción utilizando algoritmos NLP.
Formatos de exportación	TXT y JSON.
Cuestionario generado	Preguntas basadas en el contenido resumido (en evaluación).
Estadísticas del procesamiento	Datos sobre la cantidad de tokens utilizados, tiempo de ejecución y costo aproximado.

Figura 4.1: Salida del resumen

The screenshot shows a web-based questionnaire generator interface. On the left, there's a sidebar with options for generating files from YouTube videos. The main area is titled "Generador de Resúmenes y Cuestionarios a partir de Archivos o Videos de YouTube". It includes a URL input field with a sample URL, a "Generar Cuestionario" button, and a large section for generating a questionnaire from a video URL. This section contains five numbered questions with multiple-choice answers:

1. ¿Qué comando se utiliza para crear un almacenamiento (storage) en LXD que esté disponible para todo el clúster?
 - sudo lxc storage create local storage
 - sudo lxd init storage
 - sudo lxc config storage
 - sudo storage create lxd
2. ¿Cuál es el propósito de asignar perfiles de red (red interna, externa, mixta) a contenedores en LXD?
 - Controlar el acceso a recursos del sistema
 - Gestionar la memoria RAM asignada
 - Definir la configuración de red para aislamiento o conectividad
 - Establecer permisos de usuario
3. ¿Qué sucede cuando se ejecuta el comando "lxc list" y un nodo aparece como no disponible?
 - El nodo está funcionando correctamente, es un error visual
 - El nodo está temporalmente fuera de línea por mantenimiento
 - Hay un problema de permiso o configuración en el nodo
 - El nodo ha sido eliminado del clúster
4. ¿Cómo se asigna una dirección IP estática a una interfaz de red en un contenedor LXD?
 - Editando el archivo de configuración de la red del contenedor
 - Utilizando el comando "lxc config set" con la opción "network"
 - Reiniciando el servicio de red del contenedor
 - Sólo es posible mediante DHCP
5. ¿Qué diferencia a un contenedor de una máquina virtual en el contexto de LXD?
 - La cantidad de memoria RAM asignada
 - El tipo de almacenamiento utilizado
 - La presencia de un sistema operativo huésped
 - La etiqueta "vm" en el lanzamiento (launch) en lugar de "container"

At the bottom, there is a "Enviar respuestas" (Send responses) button.

Figura 4.2: Salida del cuestionario generado

Capítulo 5

Discusión y Análisis de Resultados

5.1. Costo de tokens

La cantidad de créditos que se consumen en la API de OpenAI depende de varios factores, como el modelo utilizado, la longitud del texto de entrada, la complejidad de la tarea y la configuración de la consulta. En este caso particular, los costos se determinan por los siguientes factores.

5.1.1. Costos de créditos en OpenAI

OpenAI utiliza un sistema de créditos para medir el uso de su API. Cada crédito equivale a una unidad de procesamiento de lenguaje natural. El costo de los créditos varía según el modelo y la tarea que se esté realizando.

5.1.2. Estimación por consulta

Para este caso en específico con los parámetros actuales del cuestionario de 5 preguntas a partir de un texto, podemos estimar el costo de créditos para distintos modelos de la siguiente manera:

5.1.2.1. Nemotron-70b-Instruct

- Modelo: Nvidia/Nemotron-70b-Instruct
- Costo: 10-20 créditos por 100 tokens de texto de entrada
- Longitud del texto de entrada: Aproximadamente 100-200 tokens

1. Costo de entrada: 1-4 créditos
2. Costo adicional por generación de 5 preguntas: 10-20 créditos
3. **Costo total: 11-24 créditos**

5.1.2.2. Nemoguard-8b

- Modelo: Nvidia/Nemoguard-8b
- Costo: 8-15 créditos por 100 tokens de texto de entrada
- Longitud del texto de entrada: Aproximadamente 100-200 tokens
 1. Costo de entrada: 0.8-3 créditos
 2. Costo adicional por generación de 5 preguntas: 8-15 créditos
 3. **Costo total: 8.8-18 créditos**

5.1.2.3. Qwen-32b

- Modelo: Qwen-32b
- Costo: 12-22 créditos por 100 tokens de texto de entrada
- Longitud del texto de entrada: Aproximadamente 100-200 tokens
 1. Costo de entrada: 1.2-4.4 créditos
 2. Costo adicional por generación de 5 preguntas: 12-22 créditos
 3. **Costo total: 13.2-26.4 créditos**

5.1.3. Comparación de Modelos

A continuación, se presenta un gráfico comparativo del costo total estimado para cada modelo presentes en la figura 5.1

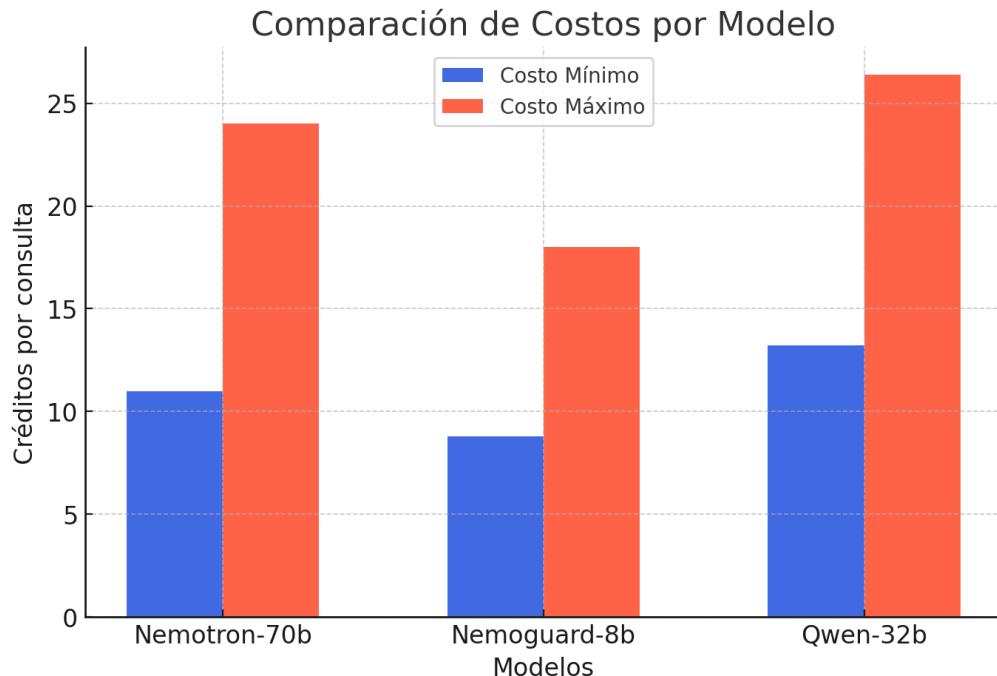


Figura 5.1: Comparación del costo por consulta entre distintos modelos.

Los resultados muestran que el modelo Nemotron-70b-Instruct es más eficiente en términos de costos y generación de preguntas en comparación con Nemoguard-8b y Qwen-32b. Aunque Nemoguard-8b presenta un costo menor, su desempeño en generación de texto es inferior. Qwen-32b, por otro lado, tiene un costo mayor sin una mejora significativa en el rendimiento frente a Nemotron-70b-Instruct. Por lo tanto, Nemotron-70b-Instruct es la opción más balanceada entre costo y rendimiento.

5.2. Optimización del Algoritmo mediante el Uso de Librerías como TextRank

En las etapas iniciales del proyecto, el proceso de extracción de oraciones relevantes se realizaba de manera manual, lo que implicaba una serie de cálculos intensivos y repetitivos. Estos cálculos incluían la **construcción de una matriz de similitud** entre oraciones, la aplicación de factores de ponderación, y la normalización de la matriz para identificar las frases más relevantes. Si bien este enfoque permitía un control detallado sobre cada paso del proceso, resultaba computacionalmente costoso, especialmente cuando se trabajaba con transcripciones que superaban las **1000 palabras**. En estos

casos, el tiempo de procesamiento aumentaba significativamente, lo que limitaba la escalabilidad del algoritmo.

Para abordar este problema, se implementó la **librería TextRank**, un algoritmo basado en grafos que automatiza y optimiza la extracción de oraciones relevantes. A diferencia del enfoque manual, TextRank no requiere la construcción explícita de una matriz de similitud ni la normalización manual de los datos. En su lugar, el algoritmo construye un grafo donde las oraciones son nodos y las aristas representan las relaciones de similitud entre ellas. La importancia de cada oración se determina mediante un proceso iterativo similar al utilizado en el algoritmo PageRank, lo que permite identificar las oraciones más relevantes de manera eficiente.

La adopción de TextRank no solo **redujo el tiempo de procesamiento**, sino que también simplificó el código y lo hizo más mantenible. En particular, se observó una mejora significativa en el rendimiento al trabajar con textos largos, donde el enfoque manual resultaba prohibitivamente lento. Además, TextRank demostró ser más robusto y escalable, permitiendo su aplicación en conjuntos de datos más grandes sin comprometer la precisión de los resultados.

En resumen, la transición de un enfoque manual a la utilización de TextRank representó una optimización clave en el proyecto. Esta mejora no solo aceleró el procesamiento de textos extensos, sino que también permitió enfocar los esfuerzos en otros aspectos del análisis, como la interpretación de resultados y la integración con otras herramientas. Este cambio subraya la importancia de aprovechar librerías especializadas para tareas complejas, especialmente en el ámbito del procesamiento de lenguaje natural, donde la eficiencia y la escalabilidad son fundamentales.

5.3. Uso de umbral en lugar de sentencias

En este estudio, se exploró la implementación de un umbral de similitud como alternativa a la definición de un número fijo de oraciones para la generación de resúmenes. Este enfoque se basa en la distribución de similitudes entre oraciones, como se muestra en la Figura 5.2, que ilustra cómo se calcula la similitud entre las oraciones y de qué parte del texto se extraen las palabras clave para el resumen.

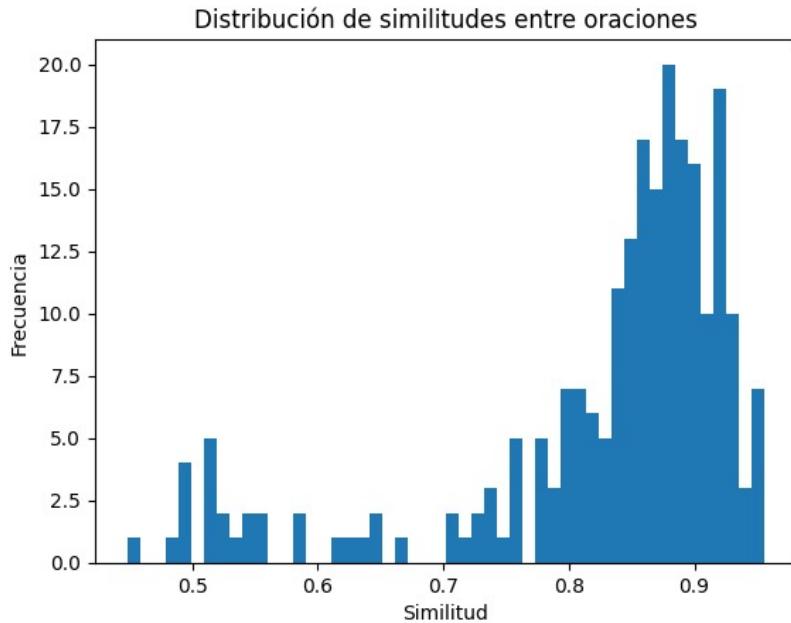


Figura 5.2: Distribución de similitudes entre oraciones.

5.3.1. Comparación entre Métodos

Método de Número Fijo de Oraciones: Este enfoque limita el resumen a un número específico de oraciones, determinado por un parámetro fijo. Por ejemplo, en el siguiente fragmento de código, se extraen las 8 oraciones más importantes según TextRank:

```

1
2 # Imprimir las oraciones m s importantes
3 print("\nOraciones m s importantes seg n TextRank:")
4 for sent in doc._.textrank.summary(limit_sentences=8):
5     print(f"\n- {sent}")

```

Este método puede ser útil para controlar la longitud del resumen, pero tiene la desventaja de que puede omitir información relevante si el número de oraciones seleccionadas es demasiado bajo.

Método de Umbral (Threshold): En contraste, el método de umbral selecciona todas las oraciones cuya similitud supera un valor predefinido. Esto se ilustra en el siguiente fragmento de código, donde se establece un umbral de 0.8 para añadir aristas basadas en similitud:

```
1
```

```
2 # Aadir aristas basadas en similitud
3 threshold = 0.8 # Ajusta el umbral aqu
4 for i in range(len(sentences)):
5     for j in range(i + 1, len(sentences)):
6         similarity = sentences[i].similarity(sentences[j])
7         if similarity > threshold:
8             G.add_edge(i, j, weight=similarity)
```

Este enfoque es más flexible y permite generar resúmenes más largos y precisos, ya que incluye todas las oraciones que cumplen con el criterio de similitud. Además, se adapta mejor a la variabilidad del texto, lo que lo hace más robusto en comparación con el método de número fijo de oraciones. Para ver el código completo dirigirse al apartado anexos en 7, en donde se encuentra todo el código de la implementación completa del algoritmo.

Capítulo 6

Conclusiones

En este trabajo se desarrolló una herramienta basada en inteligencia artificial para sintetizar información a partir de enlaces de YouTube y documentos en formatos PDF, DOCX y PPT, con el fin de generar resúmenes automáticos y cuestionarios interactivos que apoyen el aprendizaje. A continuación, se presentan las conclusiones relacionadas con los objetivos propuestos:

Se logró desarrollar una herramienta que integra técnicas de inteligencia artificial para procesar y resumir información de diversas fuentes, incluyendo videos de YouTube y documentos en diferentes formatos. Esta herramienta no solo genera resúmenes automáticos, sino que también crea cuestionarios interactivos, lo que facilita el proceso de aprendizaje.

Transcripción Automática: Se implementó un sistema de transcripción automática que extrae el texto de videos de YouTube y lo estructura en un formato utilizable para su posterior procesamiento. Este sistema demostró ser eficiente y preciso, permitiendo la integración de contenido multimedia en el proceso de generación de resúmenes.

Modelo de Procesamiento de Lenguaje Natural: Se diseñó un modelo de procesamiento de lenguaje natural que genera resúmenes automáticos a partir de los textos extraídos. Este modelo utiliza técnicas avanzadas para identificar y resumir la información más relevante, lo que mejora la calidad de los resúmenes generados.

Generación Automática de Cuestionarios: Se desarrolló un módulo que genera cuestionarios automáticamente a partir de los resúmenes, utilizando técnicas de procesamiento de lenguaje natural y la API de GPT. Este módulo permite crear preguntas interactivas que refuerzan el aprendizaje y la retención de información.

Integración en una pagina web: Se logró implementar la herramienta con todos sus algoritmos, dentro de un sitio web el cual es intuitivo para el usuario y fácil de usar.

Bibliografía

- Alonso, G., Pérez, C., & Ramos, L. (2021). Automatic quiz generation from educational texts: A Moodle-based approach. *International Journal of Artificial Intelligence in Education*, 31(2), 211-234.
- Arora, P., & Lahiri, A. (2020). Automatic Lecture Summarization Using ASR and Text Ranking. *IEEE EDUCON*.
- Chen, L., et al. (2021). AutoNotes: Generating Educational Notes from Video Transcripts in LMS. *AIED*.
- Gupta, V., & Lehal, G. S. (2019). A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*, 10(4), 345-355.
- Kurdi, M., Leo, J., Parsia, B., Sattler, U., Al-Emran, M., & Aouad, L. (2020). A systematic review of automatic question generation for educational purposes. *International Journal of Educational Technology in Higher Education*, 17(1), 1-34.
- Liu, Y., et al. (2021). Video Lecture Processing: From Speech to Knowledge. *ACM Multimedia*.
- Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, 2(2), 159-165.
- Radev, D. R., Jing, H., Styś, M., & Tam, D. (2004). Centroid-based summarization of multiple documents. *Information Processing Management*, 40(6), 919-938.
- See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 1073-1083.
- Wang, H., et al. (2022). Multimodal Video Summarization via Transformer Models. *NAACL*.

Capítulo 7

Anexos

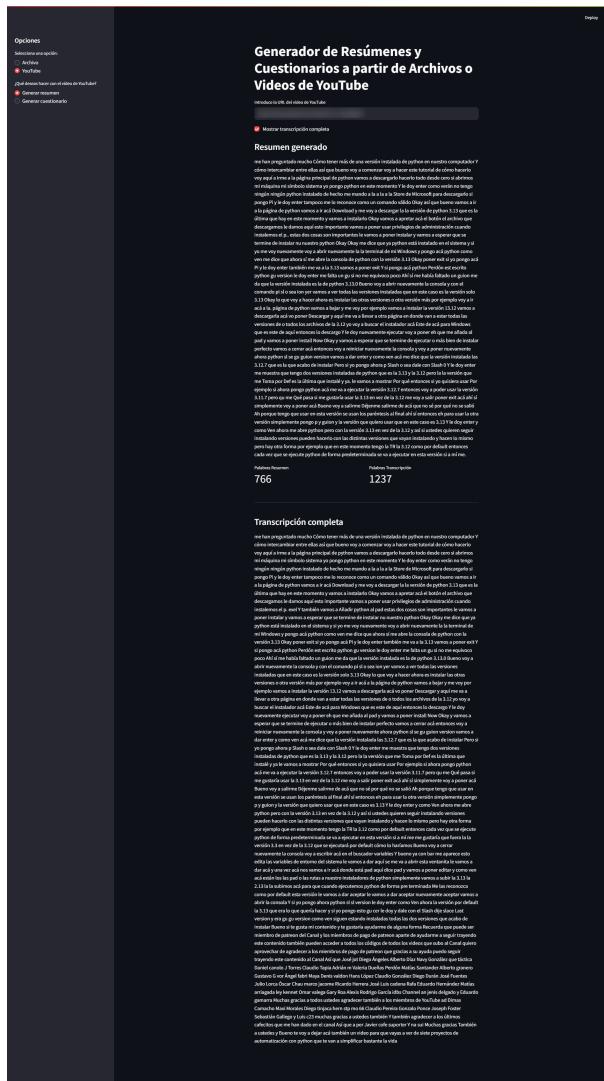


Figura 7.1: Interfaz del resumidor de videos de YouTube

Opciones:

Selecciona una opción:
 Archivo
 YouTube

¿Qué deseas hacer con el vídeo de YouTube?
 Generar resumen
 Generar cuestionario

Generador de Resúmenes y Cuestionarios a partir de Archivos o Videos de YouTube

Introduce la URL del video de YouTube

Generar Cuestionario

Cuestionario

1. ¿Por qué es útil tener múltiples versiones de Python instaladas en un computador?

Selecciona una opción:
 Para ejecutar solo la última versión disponible
 Para poder trabajar con diferentes proyectos que requieren versiones específicas de Python
 Para eliminar la necesidad de actualizar Python
 Para reducir el espacio de almacenamiento utilizado por Python

2. ¿Cómo se pueden listar todas las versiones de Python instaladas en un sistema utilizando la consola?

Selecciona una opción:
 Ejecutando `ls -al`
 Ejecutando `ls -l`
 Ejecutando `ls -l /usr/bin/python`
 Ejecutando `ls -l /usr/bin/python3`

3. ¿Cómo se selecciona una versión específica de Python para su uso por defecto en el sistema?

Selecciona una opción:
 Editando las variables de entorno del sistema y reordenando las rutas de acceso de Python
 Desinstalando otras versiones de Python excepto la deseada
 Utilizando el comando `python -m venv`
 Reinstalando la versión deseada como última opción

4. ¿Cuál es el propósito de utilizar `py -version` en la consola?

Selecciona una opción:
 Para desinstalar una versión específica de Python
 Para instalar una versión específica de Python
 Para ejecutar una versión específica de Python
 Para verificar si una versión de Python está instalada

5. ¿Por qué es importante verificar la versión de Python después de instalar una nueva versión?

Selecciona una opción:
 Para asegurarse de que la versión anterior se haya desinstalado correctamente
 Para confirmar que la nueva versión se ha instalado y configurado correctamente
 Para verificar la compatibilidad con otros software
 Para reiniciar el sistema

Enviar respuesta.

Resultados

Pregunta 1: ¿Por qué es útil tener múltiples versiones de Python instaladas en un computador?

Tu respuesta: Para poder trabajar con diferentes proyectos que requieren versiones específicas de Python

Respuesta correcta: Para poder trabajar con diferentes proyectos que requieren versiones específicas de Python

Explicación: Tener múltiples versiones de Python instaladas permite a los desarrolladores trabajar en proyectos que dependen de versiones específicas de Python, asegurando la compatibilidad y funcionalidad adecuadas.

Correcto!

Pregunta 2: ¿Cómo se pueden listar todas las versiones de Python instaladas en un sistema utilizando la consola?

Tu respuesta: Ejecutando `ls -l` o `ls -l /usr/bin`

Respuesta correcta: Ejecutando `ls -l` o `ls -l /usr/bin`

Explicación: El comando correcto para mostrar las versiones de Python instaladas es ejecutar `ls -l` o de manera equivalente `ls -l /usr/bin`, dependiendo de cómo esté configurado el sistema.

Correcto!

Pregunta 3: ¿Cómo se selecciona una versión específica de Python para su uso por defecto en el sistema?

Tu respuesta: Reinstando la versión deseada como última opción

Respuesta correcta: Editando las variables de entorno del sistema y reordenando las rutas de acceso de Python

Explicación: Para seleccionar una versión específica de Python como predeterminada, es necesario editar las variables de entorno del sistema y asegurarse de que la ruta de acceso a la versión deseada esté por encima de las demás.

Incorrecto!

Pregunta 4: ¿Cuál es el propósito de utilizar `py -version` en la consola?

Tu respuesta: Para ejecutar una versión específica de Python

Respuesta correcta: Para ejecutar una versión específica de Python

Explicación: Utilizar `py -version` o de manera equivalente `python -version` permite ejecutar una versión específica de Python, lo que es útil para proyectos que dependen de versiones particulares.

Correcto!

Pregunta 5: ¿Por qué es importante verificar la versión de Python después de instalar una nueva versión?

Tu respuesta: Para confirmar que la nueva versión se ha instalado y configurado correctamente

Respuesta correcta: Para confirmar que la nueva versión se ha instalado y configurado correctamente

Explicación: Verificar la versión de Python después de la instalación es crucial para confirmar que el proceso de instalación se completó con éxito y que la versión está lista para su uso.

Correcto!

Puntuación final: 4/5

Figura 7.2: Interfaz del generador de cuestionarios

A continuación se presenta el código fuente completo de la aplicación web desarrollada en Python, basada en Streamlit. Esta implementación permite cargar archivos o enlaces de YouTube, procesar el contenido mediante técnicas de PLN, generar resúmenes y cuestionarios educativos.

```
1 import streamlit as st
2 import PyPDF2
3 import docx
4 from pptx import Presentation
5 from youtube_transcript_api import YouTubeTranscriptApi
6 import spacy
7 import pytextrank
8 import re
9 import json
10 import networkx as nx
11 import pandas as pd
12 from openai import OpenAI
13
14 # Cargar el modelo de spaCy para español
15 nlp = spacy.load("es_core_news_md")
16 nlp.add_pipe("textrank")
17
18 # Configurar el cliente de OpenAI (NVIDIA API)
19 client = OpenAI(
20     base_url="https://integrate.api.nvidia.com/v1",
21     api_key = "your_api_key"
22)
```

Listado de código 7.1: Carga de librerías y configuración del modelo

```
1 def extract_text_from_pdf(pdf_file):
2     try:
3         text = ""
4         reader = PyPDF2.PdfReader(pdf_file)
5         for page in reader.pages:
6             text += page.extract_text()
7         return text
8     except Exception as e:
9         st.error(f"Error al procesar el archivo
10 PDF: {str(e)}")
11     return None
12
13 def extract_text_from_word(docx_file):
14     try:
15         doc = docx.Document(docx_file)
16         text = "\n".join([para.text for
17             para in doc.paragraphs])
18         return text
19     except Exception as e:
20         st.error(f"Error al procesar el archivo
21 DOCX: {str(e)}")
22     return None
23
24 def extract_text_from_ppt(ppt_file):
25     try:
26         prs = Presentation(ppt_file)
27         text = ""
28         for slide in prs.slides:
29             for shape in slide.shapes:
30                 if hasattr(shape, "text"):
31                     text += shape.text + "\n"
32     return text
33     except Exception as e:
34         st.error(f"Error al procesar el archivo
35 PPTX: {str(e)}")
36     return None
```

Listado de código 7.2: Funciones para extraer texto de PDF, Word y PowerPoint

```
1 def chunk_text(text, words_per_chunk=200):
2     words = text.split()
3     chunks = []
4     current_chunk = []
5     current_count = 0
6
7     for word in words:
8         current_chunk.append(word)
9         current_count += 1
10        if current_count >= words_per_chunk or word.endswith('.'):
11            chunks.append(''.join(current_chunk))
12            current_chunk = []
13            current_count = 0
14
15    if current_chunk:
16        chunks.append(''.join(current_chunk))
17    return chunks
18
19 def generate_summary(text, num_sentences=5):
20     text = text.replace("\n", "").replace("\r", "")
21     text = re.sub(' +', ' ', text)
22     chunks = chunk_text(text, words_per_chunk=200)
23     processed_text = ". ".join(chunks)
24     doc = nlp(processed_text)
25     sentences = list(doc.sents)
26     G = nx.Graph()
27
28     for i, sent in enumerate(sentences):
29         G.add_node(i, text=sent.text)
30
31     threshold = 0.90
32     for i in range(len(sentences)):
33         for j in range(i + 1, len(sentences)):
34             similarity = sentences[i].similarity(sentences[j])
35             if similarity > threshold:
36                 G.add_edge(i, j, weight=similarity)
37
38     important_sentences = set()
39     for i in G.nodes:
```

```
40     if G.degree[i] > 0:
41         important_sentences.add(i)
42
43     summary = [sentences[i].text for i in
44     sorted(important_sentences)]
45     return '\n'.join(summary[:num_sentences])
```

Listado de código 7.3: Funciones para segmentar texto y generar resúmenes

```
1 def count_words(text):
2     return len(text.split())
```

Listado de código 7.4: Función auxiliar para contar palabras

```
1 def generate_quiz(text):
2     try:
3         prompt = '''Genera un cuestionario de 5
4 preguntas en formato JSON. Cada pregunta debe tener:
5 - 4 opciones (a, b, c, d)
6 - 1 respuesta correcta
7 - Explicación breve de la respuesta'''
8
9         completion = client.chat.completions.create(
10             model="nvidia/llama-3.1-nemotron-70b-instruct",
11             messages=[
12                 {"role": "system", "content":
13                  "Eres una asistente que genera cuestionarios
14 educativos en formato JSON listo."},
15                 {"role": "user", "content":
16                  f"{prompt}\n\nTexto base: {text}"}
17             ],
18             temperature=0.5,
19             top_p=1,
20             max_tokens=1024,
21             stream=False
22         )
23
24         response = completion.choices[0].message.content
25         cleaned_response =
26         response[response.find('{'):response.rfind('}')+1]
27         return json.loads(cleaned_response)
28
29     except Exception as e:
30         st.error(f"Error al generar el
31 cuestionario: {str(e)}")
32         return None
```

Listado de código 7.5: Generación automática de cuestionarios con la API de NVIDIA

```
1 st.title('Generador de Resúmenes  
2 y Cuestionarios a partir de Archivos o Videos de YouTube')  
3  
4 with st.sidebar:  
5     st.header("Opciones")  
6     option = st.radio  
7         ("Selecciona una opción:", ("Archivo", "YouTube"))  
8     if option == "YouTube":  
9         youtube_option =  
10            st.radio("Quieres hacer con el video de YouTube?",  
11                ("Generar resumen", "Generar cuestionario"))
```

Listado de código 7.6: Inicio de la aplicación con Streamlit: carga e interfaz