### 

**Facultad de Negocios y Tecnologías - Campus Ixtaczoquitlán**

Región Orizaba-Córdoba

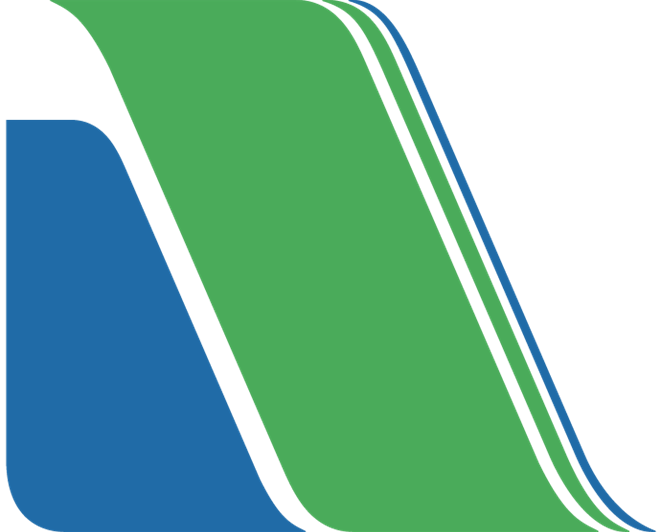
**Tecnologías de Información en las Organizaciones**

**Teclado Virtual Inteligente con Autocompletado por LSTM**

**Inteligencia Artificial**

**Presenta:**

**VAZQUEZ RUIZ RUTH**



**13 de Junio de 2025**

### **Introducción**

En las últimas décadas, el avance en las tecnologías de interacción humano-computadora ha abierto nuevas posibilidades para la comunicación con dispositivos digitales. Una de las áreas más prometedoras es la interacción basada en gestos, la cual permite controlar sistemas informáticos mediante movimientos naturales del cuerpo, como los de las manos, sin necesidad de contacto físico con periféricos tradicionales como teclados o ratones.

Este tipo de interacción es especialmente útil en contextos donde las condiciones físicas o del entorno dificultan el uso de interfaces convencionales. Por ejemplo, personas con discapacidades motrices enfrentan barreras importantes al usar teclados físicos, y en entornos clínicos o industriales donde se requieren manos libres, una interfaz gestual representa una solución funcional e higiénica.

A pesar de su potencial, los sistemas actuales de entrada por gestos todavía presentan limitaciones en cuanto a velocidad, precisión y capacidad para adaptarse al contexto del usuario. Específicamente, la ausencia de mecanismos de asistencia inteligente, como el autocompletado de texto o la predicción contextual, dificulta su uso como herramienta práctica para tareas de escritura fluida. Escribir letra por letra mediante gestos puede resultar una experiencia lenta, poco eficiente y demandante a nivel cognitivo.

### **Problemática**

Las personas con movilidad reducida experimentan dificultades significativas para interactuar con tecnologías que se apoyan en dispositivos físicos. Aunque existen soluciones como teclados en pantalla y software de reconocimiento de voz, estas alternativas no siempre son adecuadas o inclusivas. Los teclados en pantalla, por ejemplo, requieren un alto grado de precisión y estabilidad motora, mientras que los sistemas de voz pueden fallar en ambientes ruidosos o cuando el usuario presenta impedimentos del habla.

Adicionalmente, los sistemas de autocompletado disponibles en plataformas móviles o editores de texto suelen ser demasiado básicos. En general, operan sobre modelos n-gram o diccionarios predefinidos que no consideran el contexto semántico completo, el historial del usuario ni su estilo lingüístico particular. Esta falta de adaptabilidad limita su utilidad real en entornos personalizados y su capacidad de acelerar el proceso de escritura.

En este escenario, surge la necesidad de desarrollar una solución más robusta y accesible, que combine la detección de gestos con modelos de lenguaje modernos para facilitar la escritura sin contacto y de manera contextualmente inteligente.

### **Justificación**

Las personas con movilidad reducida experimentan dificultades significativas para interactuar con tecnologías que se apoyan en dispositivos físicos. Aunque existen soluciones como teclados en pantalla y software de reconocimiento de voz, estas alternativas no siempre son adecuadas o inclusivas. Los teclados en pantalla, por ejemplo, requieren un alto grado de precisión y estabilidad motora, mientras que los sistemas de voz pueden fallar en ambientes ruidosos o cuando el usuario presenta impedimentos del habla.

Adicionalmente, los sistemas de autocompletado disponibles en plataformas móviles o editores de texto suelen ser demasiado básicos. En general, operan sobre modelos n-gram o diccionarios predefinidos que no consideran el contexto semántico completo, el historial del usuario ni su estilo lingüístico particular. Esta falta de adaptabilidad limita su utilidad real en entornos personalizados y su capacidad de acelerar el proceso de escritura.

En este escenario, surge la necesidad de desarrollar una solución más robusta y accesible, que combine la detección de gestos con modelos de lenguaje modernos para facilitar la escritura sin contacto y de manera contextualmente inteligente.

**Arquitectura del Sistema**

**Tecnologías implementadas:**

* Python 3
* OpenCV y MediaPipe para detección de mano y reconocimiento de gestos.
* Pygame y PyAutoGUI para efectos de sonido e interacción con el sistema.
* TensorFlow y Keras para el modelo LSTM de autocompletado.
* Visual Studio Code como entorno de desarrollo para organizar, escribir y probar todo el código del sistema.

**Diagrama del sistema  
  
**

### **Desarrollo y Prototipo**

Con el objetivo de validar la propuesta de un teclado virtual inteligente controlado por gestos, se desarrolló un prototipo funcional que integra técnicas de visión por computadora y modelos de lenguaje basados en aprendizaje profundo. El sistema permite ingresar texto sin contacto físico, utilizando únicamente movimientos de la mano frente a una cámara, y ofrece sugerencias de autocompletado en tiempo real para mejorar la eficiencia y la experiencia del usuario.

El prototipo está estructurado en tres componentes principales, cada uno con responsabilidades específicas dentro del flujo general de funcionamiento:

* **teclado.py**: Implementa el sistema de visión por computadora encargado de detectar la mano del usuario y reconocer los gestos realizados. Específicamente, interpreta la interacción entre el dedo índice y el pulgar como una señal de selección, simulando la pulsación de una tecla sobre un teclado virtual.
* **entrenar\_modelo.py**: Se encarga del entrenamiento de un modelo de red neuronal tipo LSTM (Long Short-Term Memory) utilizando un corpus en español. Este modelo aprende a predecir el siguiente carácter con base en la secuencia de entrada, lo que permite implementar una función de autocompletado adaptativa.
* **server.py**: Actúa como servidor de predicción. Carga el modelo previamente entrenado y expone una interfaz para realizar consultas de autocompletado en tiempo real, facilitando la integración con el sistema de entrada por gestos.

El teclado virtual se renderiza en pantalla, permitiendo al usuario interactuar con él mediante gestos. Cuando se detecta que el dedo índice y el pulgar se aproximan, se interpreta como una pulsación sobre la tecla seleccionada. Las sugerencias generadas por el modelo LSTM se presentan visualmente sobre el teclado, y pueden insertarse directamente mediante una tecla especial denominada **“COMPLETAR”**, que introduce la palabra sugerida en el texto en curso.

Claro, aquí tienes una versión mejorada y más estructurada del texto. He mantenido el contenido técnico, pero con una redacción más clara, formal y coherente para su inclusión en un documento académico o técnico:

### **Implementación**

La implementación del sistema combina diversas tecnologías y bibliotecas para permitir una experiencia fluida, sin contacto, en la escritura de texto mediante gestos. Esta sección describe con mayor profundidad cada uno de los módulos y funciones desarrolladas para lograr una interacción efectiva con el teclado virtual y habilitar la funcionalidad de autocompletado inteligente.

### **6.1. Captura y Detección de Gestos**

La captura de los gestos del usuario se realiza en tiempo real utilizando técnicas de visión por computadora. La detección precisa de la mano es fundamental para que el sistema reconozca los movimientos intencionados y los traduzca en acciones concretas dentro del teclado virtual.

#### **MediaPipe Hands**

Se utiliza la solución **MediaPipe Hands**, una biblioteca desarrollada por Google, que permite detectar con alta precisión la posición de las manos en una imagen o video en tiempo real. Esta herramienta identifica 21 puntos clave (landmarks) en cada mano, que incluyen las articulaciones y puntas de los dedos. Estos puntos se obtienen en coordenadas normalizadas y son esenciales para interpretar gestos específicos del usuario.

#### **distance()**

Esta función calcula la **distancia euclidiana** entre dos puntos clave: la punta del **dedo índice** y la del **pulgar**. Cuando esta distancia cae por debajo de un umbral definido (por ejemplo, 30 píxeles), el sistema interpreta que el usuario está simulando una pulsación sobre una tecla. Este gesto se considera una acción de **“clic virtual”**, y es la base para permitir la selección de caracteres sin contacto físico.

### **6.2. Interacción con el Teclado Virtual**

Una vez detectados los gestos, el siguiente paso es permitir que el usuario interactúe con una interfaz de teclado renderizada visualmente en pantalla.

#### **draw\_keyboard()**

Esta función se encarga de **dibujar el teclado virtual** en la ventana de salida de video. Define la disposición de las teclas, su forma (generalmente rectangular), su etiqueta (letra o símbolo) y su posición relativa. Además, si el dedo índice se encuentra sobre alguna tecla, esta se resalta gráficamente (por ejemplo, cambiando de color), lo que proporciona una **retroalimentación visual inmediata** que ayuda al usuario a confirmar su intención de selección.

#### **Lógica de “clic” virtual**

El sistema evalúa continuamente si el dedo índice está superpuesto a una tecla. Si además detecta que se aproxima al pulgar (mediante distance()), se ejecuta una acción de **selección automática**. Esta lógica reemplaza la acción de presionar una tecla física, permitiendo que el gesto sea interpretado como una entrada válida en el sistema.

### **6.3. Gestión del Texto Ingresado**

Para registrar el texto que el usuario va construyendo y controlar el sistema de sugerencias, se utilizan variables específicas en tiempo de ejecución.

#### **texto\_escrito**

Es una variable que actúa como **buffer de entrada**, donde se van concatenando los caracteres seleccionados por el usuario. Este texto es utilizado tanto para mostrar en pantalla el progreso de escritura como para alimentar el módulo de autocompletado.

#### **sugerencia**

Esta variable almacena la **palabra sugerida más reciente**, obtenida ya sea mediante el sistema básico de búsqueda o a través del modelo LSTM. Cuando el usuario presiona la tecla especial "COMPLETAR", el contenido de esta variable se inserta automáticamente al final de texto\_escrito, reemplazando la palabra parcial que estaba siendo escrita.

### **6.4. Visualización y Retroalimentación**

Una parte esencial de la experiencia de usuario es recibir información visual y auditiva que indique que el sistema está respondiendo correctamente a sus acciones.

#### **cv2.putText() y cv2.rectangle()**

Estas funciones, pertenecientes a la biblioteca **OpenCV**, permiten **renderizar texto y formas** directamente sobre los fotogramas de video. Se utilizan para:

* Mostrar en pantalla el texto que se ha ingresado (texto\_escrito).
* Delimitar las teclas del teclado virtual.
* Resaltar teclas activas o seleccionadas.
* Visualizar sugerencias generadas por el sistema de autocompletado.

#### **pygame.mixer.Sound**

Se utiliza la biblioteca **Pygame** para añadir **retroalimentación auditiva** al sistema. Cada vez que se detecta una selección válida de tecla, se reproduce un sonido corto (similar a un clic). Esto refuerza la experiencia interactiva y permite al usuario saber que su acción fue registrada, incluso si no está mirando directamente la pantalla.

### **6.5. Sistema de Autocompletado**

El sistema de autocompletado es un componente clave del prototipo. Permite aumentar significativamente la velocidad de escritura al ofrecer sugerencias inteligentes basadas en el contexto del texto ingresado.

#### **autocomplete\_simple()**

Es una función que realiza una **búsqueda básica** de coincidencias en un corpus de texto. Utiliza la última palabra escrita por el usuario como prefijo para encontrar posibles palabras completas que comienzan igual. Esta funcionalidad es útil en situaciones donde el modelo LSTM no está disponible o como solución de respaldo más rápida y ligera.

#### **predict\_next\_char()**

Este método forma parte del archivo **server.py** y actúa como interfaz para interactuar con el modelo LSTM previamente entrenado. Dado un contexto textual, el modelo predice el siguiente carácter más probable que debería seguir en la secuencia. De esta forma, el sistema puede construir palabras letra por letra hasta obtener una sugerencia completa. Esta estrategia ofrece una alternativa más poderosa y personalizada al autocompletado básico, ya que aprende patrones del lenguaje y del estilo del usuario a lo largo del tiempo.

Claro, te voy a desarrollar cada una de esas pruebas con explicación extensa, técnica y clara para que puedas incluirlas en tu documento y quede totalmente justificado y comprensible. Vamos una por una.

## **Prueba 1: Detección de gestos (clic virtual)**

### **Objetivo de la prueba**

Esta prueba tiene como finalidad evaluar la capacidad del sistema para detectar correctamente el gesto que simula la pulsación de una tecla virtual. En nuestro proyecto, el gesto se define como la proximidad entre la punta del dedo índice y el pulgar, y se interpreta como un “clic virtual”.

### **Descripción**

La detección del clic virtual es el punto crítico que permite la interacción con el teclado sin contacto físico. Utilizando el módulo **MediaPipe Hands**, el sistema rastrea en tiempo real las coordenadas de 21 puntos clave de la mano, especialmente los dedos índice y pulgar.

Para interpretar un clic, se calcula la distancia euclidiana entre estos dos puntos. Cuando la distancia es menor que un umbral establecido (por ejemplo, 2 cm en coordenadas normalizadas), el sistema reconoce que el usuario ha hecho el gesto de presionar.

### **Método de prueba**

Se llevan a cabo múltiples intentos controlados con usuarios, o con el desarrollador, en los cuales se realiza intencionadamente el gesto de clic. Se registran:

* **Clics reconocidos correctamente**: casos donde el sistema detecta el gesto cuando se hace.
* **Falsos negativos**: casos donde el usuario hace el gesto, pero el sistema no detecta ningún clic.
* **Falsos positivos**: casos donde el sistema detecta un clic sin que el usuario haya hecho el gesto.

El objetivo es maximizar los clics correctos y minimizar los errores.

### **Métricas y resultados esperados**

* **Precisión de detección**: porcentaje de clics reales detectados correctamente. Se espera alcanzar al menos un 90% para que la experiencia sea fluida.
* **Tasa de error**: número de falsos positivos y negativos, idealmente muy bajos para evitar frustración.

### **Importancia**

Sin una detección de clic fiable, la interacción con el teclado virtual no sería posible. Una tasa elevada de errores causaría que el usuario pierda confianza y se frustre, además de afectar la velocidad y precisión al escribir.

## **Prueba 2: Selección de teclas en el teclado virtual**

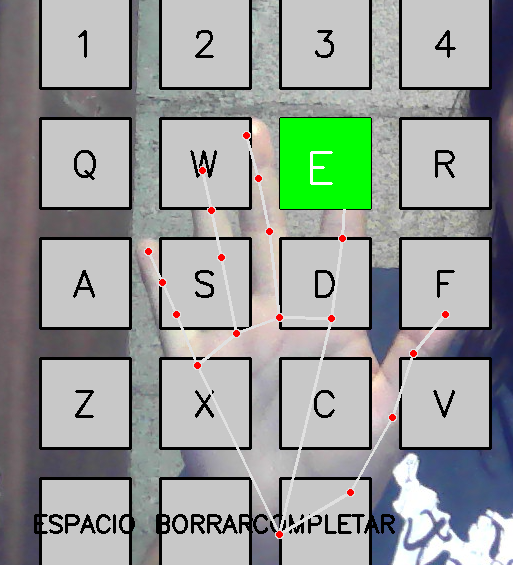
### **Objetivo de la prueba**

Esta prueba busca verificar que el sistema identifique correctamente qué tecla está siendo seleccionada por el usuario en función de la posición del dedo índice sobre el teclado virtual proyectado en pantalla.

### **Descripción**

El teclado virtual está representado gráficamente con una matriz de teclas, cada una con coordenadas y dimensiones definidas. La función draw\_keyboard() no solo renderiza las teclas, sino que también calcula en tiempo real qué tecla se encuentra bajo la posición del dedo índice.

Cuando el sistema detecta un clic virtual (proximidad índice-pulgar), se registra como una pulsación en la tecla que estaba “seleccionada” (sobre la que está apuntando el dedo índice).



### **Método de prueba**

Para validar esta funcionalidad, se realizan pruebas donde:

* El usuario mueve el dedo índice hacia diferentes teclas.
* El sistema debe actualizar la selección visualmente (resaltando la tecla).
* Al hacer el gesto de clic, debe insertar el carácter correspondiente de la tecla resaltada.

Se verifica:

* Que la tecla resaltada corresponde a la posición real del dedo índice.
* Que al hacer clic, la tecla seleccionada sea la correcta.

### **Métricas y resultados esperados**

* **Tasa de coincidencia**: porcentaje de veces que la tecla detectada coincide con la tecla apuntada.
* **Latencia de actualización**: tiempo que tarda el sistema en cambiar la tecla resaltada al mover el dedo.

Una tasa alta (por encima del 95%) y una latencia baja (menos de 100 ms) son ideales para ofrecer una experiencia de usuario fluida y natural.

### **Importancia**

Una selección incorrecta o retrasada puede hacer que el usuario escriba caracteres equivocados, lo que reduce la eficiencia y genera frustración. La retroalimentación visual inmediata y precisa es clave para que el usuario confíe en el sistema.

**Prueba 3: Escritura correcta de texto**

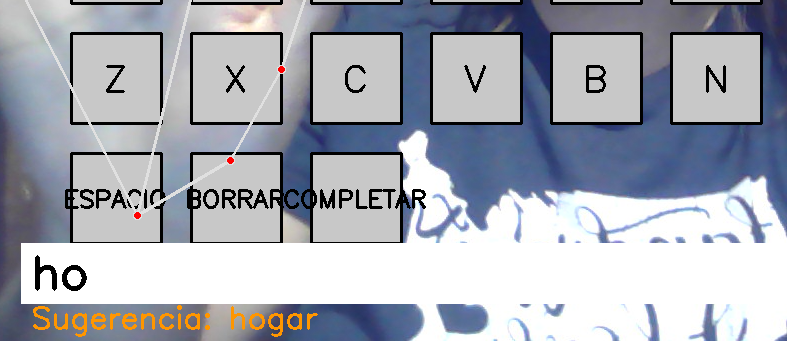
### **Objetivo de la prueba**

Esta prueba está destinada a verificar que el sistema registre de forma correcta y secuencial los caracteres seleccionados por el usuario, construyendo así el texto final que se visualiza en pantalla.

### **Descripción**

Cuando el sistema detecta un clic virtual sobre una tecla, el carácter correspondiente debe ser agregado a la variable texto\_escrito, que almacena el texto que el usuario va ingresando en tiempo real.

El texto se muestra continuamente en pantalla utilizando funciones gráficas (cv2.putText()), permitiendo al usuario observar el progreso de la escritura.



### **Método de prueba**

* Se seleccionan frases o secuencias de caracteres para que el usuario intente escribirlas mediante gestos.
* Se compara el texto final generado con el texto esperado.
* Se registran errores de inserción (caracteres incorrectos o faltantes).

Se pueden realizar pruebas con diferentes usuarios y en condiciones variables para evaluar la robustez.

### **Métricas y resultados esperados**

* **Tasa de exactitud de texto**: porcentaje de caracteres escritos correctamente respecto al texto objetivo.
* **Número de correcciones necesarias**: cuando el usuario tiene que borrar o corregir errores.

Se espera una tasa de exactitud superior al 90% para considerar que la escritura es eficaz.

### **Importancia**

El propósito fundamental del teclado virtual es permitir la entrada de texto. Por ello, garantizar la correcta escritura es vital para que el sistema sea útil en escenarios reales.

**Prueba 4: Inserción de sugerencias con la tecla “COMPLETAR”**

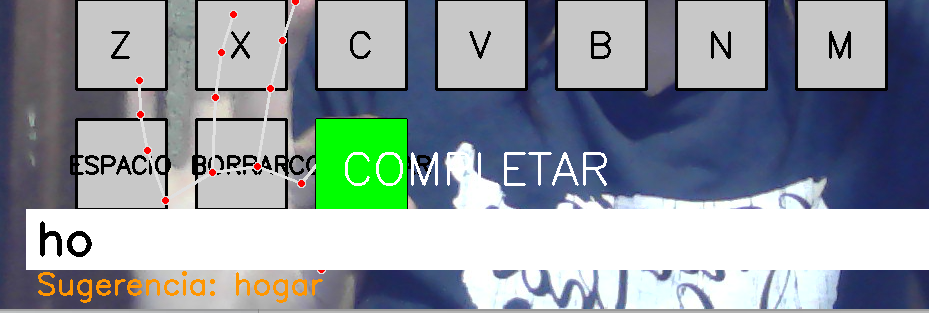
### **Objetivo de la prueba**

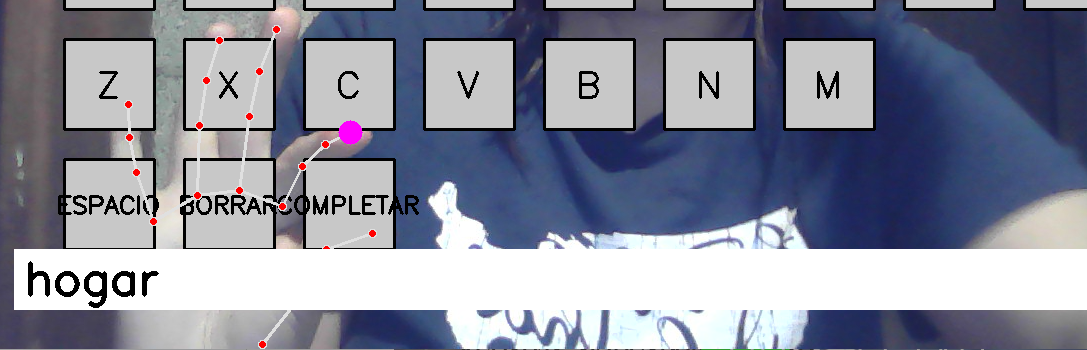
Validar que la función de autocompletado permita insertar correctamente la palabra sugerida por el modelo LSTM cuando el usuario lo desea, mejorando la velocidad y eficiencia de la escritura.

### **Descripción**

El sistema genera sugerencias de palabras basadas en el texto parcial ingresado. Estas se muestran en pantalla y el usuario puede elegir insertarlas directamente al pulsar la tecla especial “COMPLETAR” en el teclado virtual.

Cuando se activa esta función, la palabra sugerida se añade íntegramente a texto\_escrito, sustituyendo o completando la palabra actual en construcción.





### **Método de prueba**

* Se simula o se realiza la escritura de palabras incompletas.
* Se verifica que las sugerencias visualizadas sean relevantes y correctas.
* Se ejecuta el gesto de clic sobre “COMPLETAR”.
* Se verifica que la palabra sugerida se inserte correctamente en el texto.
* Se valida que el sistema actualice el texto y la interfaz adecuadamente.

### **Métricas y resultados esperados**

* **Tasa de inserción exitosa**: porcentaje de veces que la palabra sugerida se inserta correctamente.
* **Relevancia de sugerencias**: evaluación subjetiva o cuantitativa de qué tan útiles y correctas son las palabras propuestas.
* **Reducción de gestos necesarios**: comparación del número total de selecciones entre escribir sin y con autocompletado.

### **Importancia**

El autocompletado es un elemento clave para reducir la fatiga y el tiempo necesario para escribir con gestos. La correcta inserción de sugerencias es vital para que esta funcionalidad cumpla su objetivo de hacer la escritura más rápida y menos demandante.

**Conclusión**

Este proyecto evidencia la viabilidad y el potencial de combinar tecnologías de visión por computadora con modelos de lenguaje avanzado para desarrollar interfaces de interacción humano-máquina más intuitivas, accesibles e inteligentes. La implementación del modelo LSTM para autocompletado ha mostrado resultados prometedores en términos de precisión y relevancia de las sugerencias, aunque aún no se encuentra plenamente integrado en tiempo real con el teclado virtual.

Entre las limitaciones identificadas destacan la necesidad de aumentar la robustez y precisión en la detección y seguimiento de la mano, así como la optimización del flujo de interacción para que el modelo LSTM funcione como un asistente de escritura activo y fluido dentro del sistema.

**Líneas de mejora y trabajo futuro**

* Integrar completamente el modelo LSTM en el módulo principal de interacción (teclado.py), permitiendo que las sugerencias se actualicen y presenten de forma dinámica durante la escritura.
* Ampliar y diversificar el corpus de entrenamiento, además de ajustar los hiperparámetros del modelo, para mejorar la adaptabilidad y la precisión contextual del autocompletado.
* Desarrollar una interfaz de usuario más amigable, personalizable y accesible, que facilite la configuración y el uso del sistema por parte de usuarios con diferentes necesidades.
* Explorar la incorporación de soporte multilingüe y la integración de comandos por voz, ampliando las modalidades de interacción y aumentando la versatilidad del sistema.

Este proyecto sienta las bases para futuras investigaciones y desarrollos orientados a crear tecnologías inclusivas que faciliten la comunicación y la productividad de personas con movilidad reducida, y que a la vez impulsen nuevas formas de interacción natural con dispositivos digitales.