Memoria

Índice:

Roberto Frías Bernal y Sergio Roselló Morell

Idea de proyecto:

Programa que se ejecuta en segundo plano y está constantemente monitoreando lo que dices. Cuando detecta una palabra malsonante, se dá cuenta y bloquea el ordenador.

Inconvenientes de la idea de proyecto:

Tras investigar qué API's existen con capacidad text-to-speech, descubrimos que la mayoría estaba aún en desarrollo. Parece que Google cloud speech API es la más novedosa y la que más gente está adoptando. Además, es bastante sencilla de utilizar.

Elección de API (Application programming interface)

Posibles API's:

1. Google cloud speech API
2. Pocketsphinx
3. IBM speech to text (watson-developer-cloud)
4. WIT.ai

Google cloud speech API

Ventajas:

1. Integración sencilla, por tanto, podemos invertir más tiempo en desarrollar la aplicación.
2. La API nos proporciona un flag de stopword (palabra malsonante) al que podemos monitorizar facilmente y cuando ese flag cambie de False a True, simplemente, procedemos a bloquear el ordenador a nuestro usuario.

Inconvenientes:

1. El proceso de traducción lo hace en la nube (sus servidores) por tanto, nuestra app no podría funcionar offline.
2. Tenemos que estar constantemente enviando la información que recolectamos a google para que la procese.
3. De pago si llegas a ciertos niveles de uso/minuto.

Pocketsphinx

Ventajas:

1. API fuertemente integrada en el lenguaje de desarrollo seleccionado.
2. Funciona offline, por tanto nuestro software no es dependiente de que exista una conexión a internet.
3. No depende de ningun servicio externo. No creamos dependencias con servicios como Google speech API.

Inconvenientes:

1. Nosotros tenemos que lidiar con la traducción de habla a texto (Descargando e instalando diccionarios, modelos de lenguaje etc…
2. La documentación de la API es escasa.
3. Al parecer, están portando la página web a otro servicio (no disponemos de los enlaces a la documentación (temporalmente))
4. Incluso existiendo comunidad, nadie responde a las preguntas más simples

IBM speech to text

Ventajas:

1. Buena documentación
2. IBM se encarga de realizar la traducción.
3. Proporciona una opción de streaming (con algunas limitaciones)

Inconvenientes:

1. Tenemos que estár conectados a internet.
2. Creamos una dependencia con El servicio de IBM
3. La opción de streaming tiene que dividir la información para poder procesarla

WIT.ai

Ventajas:

1. Framework super moderno utilizado por un montón de empresas actualmente.
2. Ofrece muchas ventajas al programador como interfaz gráfica (web)
3. Excelente documentación

Inconvenientes:

1. El framework no se adecua exactamente a lo que queremos nosotros.
2. Para poder utilizarlo satisfactoriemente, necesitamos invertir tiempo en entrenar al sistema para que identifique las palabras que queremos.

¿Que API usamos?

Trás tener todos estos factores en mente, decidimos usar Google cloud speech API porque nuestra intención no es crear un modelo de lenguaje, (cosa que desgraciadamente tendremos que crear más adelante) sino hacer una app que te bloquee el ordenador. Si existe ya un sistema bastante bueno y además nos podemos evitar hacer la parte de análisis de datos, mejor.

Transcurso del proyecto:

Todo va sobre ruedas.

Durante dos semanas, avanzamos satisfactoriamente. Conseguimos implementar las llamadas al servicio de Google para que nos hiciese la traducción de audio a texto y averiguamos como funcionaba la API para poder hacer cosas más avanzadas. Uno de los requisitos de la aplicación es que el procesamiento de lenguaje en streaming dure al menos media hora, que esta API cumplia cuando nos decidimos por ella pero que sin aviso cambiaron a un minuto, por tanto, debíamos editar nuestro código para que llamara al servicio de Google cada minuto.

Las cosas se empiezan a complicar.

Trás implementar satisfactoriamente que nuestra aplicación nos escuche durante media hora, decidimos cambiar el lenguaje de ingles a español, que a priori es tán sencillo como cambiar una variable de en-us a es-es en la petición de traducción que enviamos a Google. Cuando hicimos este pequeño cambio, dejó de funcionar correctamente la llamada a la API. El error estaba relacionado con los caracteres especiales (fuera de la tabla ASCII) que tiene la lengua española, como por ejemplo la ñ o la á. La petición de traducción la

realizaba correctamente pero no parseaba (traducía de JSON a lenguaje natural) correctamente. Sabiendo que el fallo podía venir de dos sitios, el programa que usa la API de google para parsear la respuesta o nuestro ordenador, comprobamos que nuestro ordenador tuviera al día todos los drivers y programas necesarios para representar el texto en español y comprobamos que efectivamente, en nuestro ordenador, todo estaba al dia. Solo podemos concluír que el fallo venía de el programa que usa la API para parsear la información recibida. Es bastante posible que en el estado en el que estaba la API (beta) existiese ese fallo.

Pasamos de Google cloud speech API a pocketsphinx.

Teniendo en cuenta que este proyecto es “time-sensitive” nos vemos obligados a usar otro framework que nos permita desarrollar la aplicación.

Decidimos usar pocketsphinx, una variante de GNUsphinx más ligera, pensada para dispositivos móviles más que ordenadores pero que por su simplicidad y capacidad de hacer lo que necesitamos, nos venía bien. Como aparece en la lista de inconvenientes, pocketsphinx no es un servicio on-line por tanto, debemos implementar nosotros el proceso de traducción con las herramientas que encontramos en internet. Por defecto, el programa trae los archivos necesarios para realizar la traducción en Inglés. Para comprobar que la API funcionaba y eramos capaces de utilizarla, probamos con la version por defecto que trae. Si funciona. Esto tiene futuro, por tanto actualizamos dependencias, montamos el sistema de control de versiones y procedemos a pasarnos a pocketsphinx, averiguando donde está la documentación y entendiendo más profundamente como funciona la API.

Pasamos pocketsphinx de inglés a español.

Como vimos que nos gustaba como hacía esta API las traducciones de lengua hablada a escrita, decidimos continuar desarrollando con ella. Para hacer eso, debíamos pasar de reconocer y traducir el inglés a el español. Esto que en la API de Google es tán simple, aquí se nos está haciendo cuesta arriba. En principio, el proceso debería ser tan sencillo como descargar unos ficheros (los diccionarios de la lengua española y los modelos de lenguaje, además de archivos no tan importantes y opcionales) y colocarlos en un directorio específico, que es donde la API busca los distintos lenguajes instalados.

Las cosas se vuelven a complicar.

Algunos de los inconvenientes que hemos tenido: Pese a tener la última versión de la API (0.1.3) descargada, todas las preguntas relacionadas con los problemas que teníamos, estaban planteadas sobre unas cuantas versiones más antiguas. (0.0.9) Otro de los inconvenientes que hemos tenido es que estaban pasando a otro dominio la página web oficial de la API y por tanto, no todos los enlaces estaban disponibles. Sabemos que esto es verídico porque en la sección “contact” de la página oficial, proporcionan un grupo

de Telegram abierto en el que precisamente, la gente se quejaba de que no podía acceder a la documentación.

Lidiando con el problema.

Finalmente, en sourceforge, encontramos los archivos necessarios para poder realizar la traducción. (diccionario, modelo de lenguaje y otros archivos menos relevantes) Descargamos el paquete y lo descomprimimos en el directorio indicado, que por supuesto, no funciona. Buscamos en internet las posibles causas de nuestro problema. Pueden venir de diversos puntos. Uno de ellos, que la carpeta esté estructurada de manera inadecuada, otro que esté incorrectamente nombrada, (una de las cosas que correjimos) otra, un poco más significativo, (que también nos pasó) es que la carpeta oficial que dá soporte a la lengua española esté corrompida y falten archivos.

Conclusión y lecciones aprendidas.

Trás este fiasco, podemos concluír que implementar una API no es tán facil como parece, más aún si está en desarrollo activo (caso de Google cloud speech API) o en desarrollo peculiar (caso de pocketsphinx)

Pese a que el proyecto no ha salido como nos gustaría, hemos aprendido muchas cosas, entre otras, a usar API’s en desarrollo, (no recomendado) a gestionar un proyecto por nuestra cuenta. Hemos descubierto que podemos utilizar nuestros conocimientos como desarrolladores de software para hacer proyectos realmente creativos.