**HERRAMIENTA SPHINX4**

**Introducción**

Sphinx4 es una biblioteca de reconocimiento de voz pura de Java. Proporciona una API rápida y fácil para convertir las grabaciones de voz en texto con la ayuda de los modelos acústicos de CMUSphinx. Puede ser utilizado en servidores y en aplicaciones de escritorio. Además del reconocimiento de voz, Sphinx4 ayuda a identificar los altavoces, a adaptar los modelos, a alinear la transcripción existente con el audio para el sello de tiempo y más.

Sphinx4 es compatible con el inglés de EE. UU. Y muchos otros idiomas.

Esta aplicación es la encargada del reconocimiento de voz, la decodificación y trasformada en texto .Para eso se requiere de una gramática que nos permita a la computadora como es que el usuario va hablar, esto permite tener menos errores al momento de codificar la voz y pasarla a texto. Para pasarla a texto primero la decodificación y la compra con la gramática.

**Modelos**

De acuerdo con la estructura del habla, se utilizan tres modelos en el reconocimiento de voz para hacer la búsqueda:

* Un modelo acústico contiene propiedades acústicas para cada senone. Hay modelos independientes del contexto que contienen propiedades (los vectores de características más probables para cada fon) y dependientes del contexto (construidos desde senones con contexto).
* Un diccionario fonético contiene un mapeo de palabras a fonemas. Este mapeo no es muy efectivo. Por ejemplo, solo se observan dos o tres variantes de pronunciación. Sin embargo, es lo suficientemente práctico la mayor parte del tiempo. El diccionario no es el único método para asignar palabras a los fonemas. También podría usar alguna función compleja aprendida con un algoritmo de aprendizaje automático.
* Un modelo de lenguaje se usa para restringir la búsqueda de palabras. Define qué palabra podría seguir palabras previamente reconocidas (recuerde que la coincidencia es un proceso secuencial) y ayuda a restringir significativamente el proceso de emparejamiento eliminando palabras que no son probables. Los modelos de lenguaje más comunes son los modelos de lenguaje n-gram, que contienen estadísticas de secuencias de palabras y modelos finitos de lenguaje de estado, que definen las secuencias del habla mediante la automatización de estado finito, a veces con ponderaciones. Para alcanzar una buena tasa de precisión, su modelo de lenguaje debe tener mucho éxito en la restricción del espacio de búsqueda. Esto significa que debería ser muy bueno para predecir la próxima palabra. Un modelo de lenguaje generalmente restringe el vocabulario que se considera a las palabras que contiene. Ese es un problema para el reconocimiento de nombres. Para tratar con esto, un modelo de lenguaje puede contener fragmentos más pequeños como subpalabras o incluso fonemas. Se debe tener en cuenta que la restricción de espacio de búsqueda en este caso suele ser peor y las precisiones de reconocimiento correspondientes son menores que con un modelo de lenguaje basado en palabras.

Esas tres entidades se combinan juntas en el motor para reconocer el habla.

**Descripción general del kit de herramientas de CMUSphinx**

El kit de herramientas CMUSphinx es un kit de herramientas de reconocimiento de voz líder con varias herramientas utilizadas para crear aplicaciones de voz. CMUSphinx contiene una serie de paquetes para diferentes tareas y aplicaciones. A veces, es confuso qué elegir. Para arrojar algo de luz sobre las partes del kit de herramientas, aquí hay una lista:

* Pocketsphinx - biblioteca de reconocimiento ligera escrita en C.
* Sphinxbase - biblioteca de soporte requerida por Pocketsphinx
* Sphinx4 - reconocedor ajustable y modificable escrito en Java
* Sphinxtrain - herramientas de entrenamiento modelo acústico

**VERSIONES SITEMAS DE SOFWARE CMU-SPHINX**

**Características**

Sphinx-2

Está orientado a brindar el mejor rendimiento posible en tiempo real, sacrificando un poco la precisión. Está escrito en lenguaje de programación C.

Sphinx-3:

Está orientado a brindar la mayor precisión posible con vocabularios extensos. Inicialmente fue pensado para el procesamiento por lotes. Luego, con la aparición de mejores procesadores y memorias más económicas, se utilizá para ofrecer un reconocimiento de voz con vocabulario extenso en tiempo real con un rendimiento muy aceptable. Está escrito en lenguaje de programación C.

Sphinx-4:

Tiene las mismas características que Sphinx-3, pero esa totalmente escrito en lenguaje de programación Java. Se debe tener en cuenta que es una versión beta, por lo cual aunque se mantiene estable aun tiene varios inconvenientes. Para probarlo o integrarlo a alguna aplicación, la versión a utilizar dependerá de la conveniencia pues si la aplicación está escrita en C o C++ es mucho más conveniente acoplarle un motor en C. Si está escrita en Java por ende se usará

Sphinx-4. Las versiones de Sphinx escritas en C ofrecen un rendimiento muy superior a las escritas en Java. Ahora bien la integración de Sphinx-4 es un poco más sencilla, pues como esta construido a base de clases es más simple de comprender y aprovechar. Esta última requiere de mucha mayor capacidad de procesamiento y aproximadamente 1 Gb de memoria para ofrecer un rendimiento conveniente para realizar un reconocimiento de voz en tiempo real. Para máquinas no tan poderosas es conveniente la versión escrita en C. Sphinx puede trabaja en modo cliente servidor, por lo cual un cliente puede solicitar servicios de reconocimiento de voz a un servidor que se esta ejecutando en otro host. Esto es muy apropiado para el desarrollo de aplicaciones. De manera muy similar a un motor de bases de datos ejecutándose en un servidor independiente del cliente, así también se puede hacer con el motor de reconocimiento de voz de Sphinx. Por otro lado se tiene el módulo de entrenamiento SphinxTrain para generar los modelos que luego se pueden cargar en diferentes versiones de Sphinx

Es un framework flexible, modular y fácil de incorporar a otras aplicaciones. De esta manera, ha ayudado a fomentar las innovaciones vinculadas a la investigación de los sistemas de reconocimiento, basados en los Modelos Ocultos de Markov.

Fue desarrollado en Java, lo cual hace muy práctico el desarrollo de aplicaciones multiplataformas. Esto es uno de los cambios más importante que tiene Sphinx-4 respecto a versiones anteriores. Representa uno de los sistemas de reconocimientos de voz más completo, sólido y estable de la actualidad. Incluye muchas maneras diferentes de resolver un mismo problema, lo que lo convierte en un software flexible, capaz de ser utilizado en cualquier situación, desde aplicaciones para personas con algún tipo de discapacidad hasta laboratorios de investigación o dentro de un automóvil. Esta variedad de aplicaciones hace que sea un software altamente adaptable a cualquier ambiente.

Esto es una ventaja respecto a HTK (Hidden Markov Model Toolkit) , uno de los frameworks de reconocimiento de voz más consolidadas para los estudios universitarios. HTK ha sido programado en C, esto podría ser una desventaja a la hora de tener que exportar el software producido a una plataforma diferente de la utilizada en el desarrollo. Por eso Sphinx-4, al ser Java un lenguaje interpretado, y no un compilado como C, no se tiene necesidad de compilar un código para una plataforma específica. La mera utilización de Java como lenguaje de programación.

Ventajas

La gran variedad de APIs que existen para el desarrollo en lenguaje Java, reduce el tiempo de codificación.

El soporte integrado para multi-thread hace más sencilla la división de tareas en varios hilos de ejecución.

El Garbage Collector, que ayuda a que los programadores se concentren en el desarrollo del algoritmo y se olviden del manejo dinámico de memoria.

Otra ventaja de este framework es que es un software libre, lo cual evita el problema de tener que desarrollar un sistema completo desde cero. Además posee una estructura modular y de aplicación conectable que incorpora los patrones de diseño de sistemas existentes. En resumen, es libre para ser usado y mejorado y su plataforma de trabajo se amplió debido a la tecnología Java. Una última observación que se debe tener en cuenta en el marco de esta tesina, es que el desarrollo planteado será un prototipo de aplicación de software libre independiente de la plataforma, porque se ejecuta en una máquina virtual Java, pero la aplicación de la máquina virtual puede no ser libre.

**Desventajas**

La plataforma Java puede ocasionar problemas con la cantidad de memoria principal que un programa utiliza o referencia mientras se está ejecutando.

También relacionado con la memoria, algunos motores de voz, acceden directamente a memoria para optimizar el rendimiento durante la decodificación. Con este lenguaje, el acceso directo a memoria no está permitido. A pesar de esto, se han obtenido buenos resultados en el desarrollo y el rendimiento de aplicaciones de reconocimiento de voz. Este framework es un sistema basado en los Modelos Ocultos de Markov (HMM, Hidden Markov Model), y como todos los demás sistemas de reconocimiento de voz, para su funcionamiento primero debe aprender las características (o parámetros) de un conjunto de unidades de sonido, y luego utilizar lo que ha aprendido de estas unidades para encontrar la secuencia de unidades de sonido más probable para una señal de voz dada. El proceso de aprendizaje sobre las unidades de sonido se llama entrenamiento. El proceso de usar el conocimiento adquirido para deducir la secuencia de sonido más probable en una señal dada se llama decodificación, o simplemente reconocimiento. En consecuencia, se necesitan dichos componentes del sistema Sphinx que se pueden usar para el entrenamiento y reconocimiento. En otras palabras, son necesarios el SPHINX Trainer y el SPHINX Decoder.

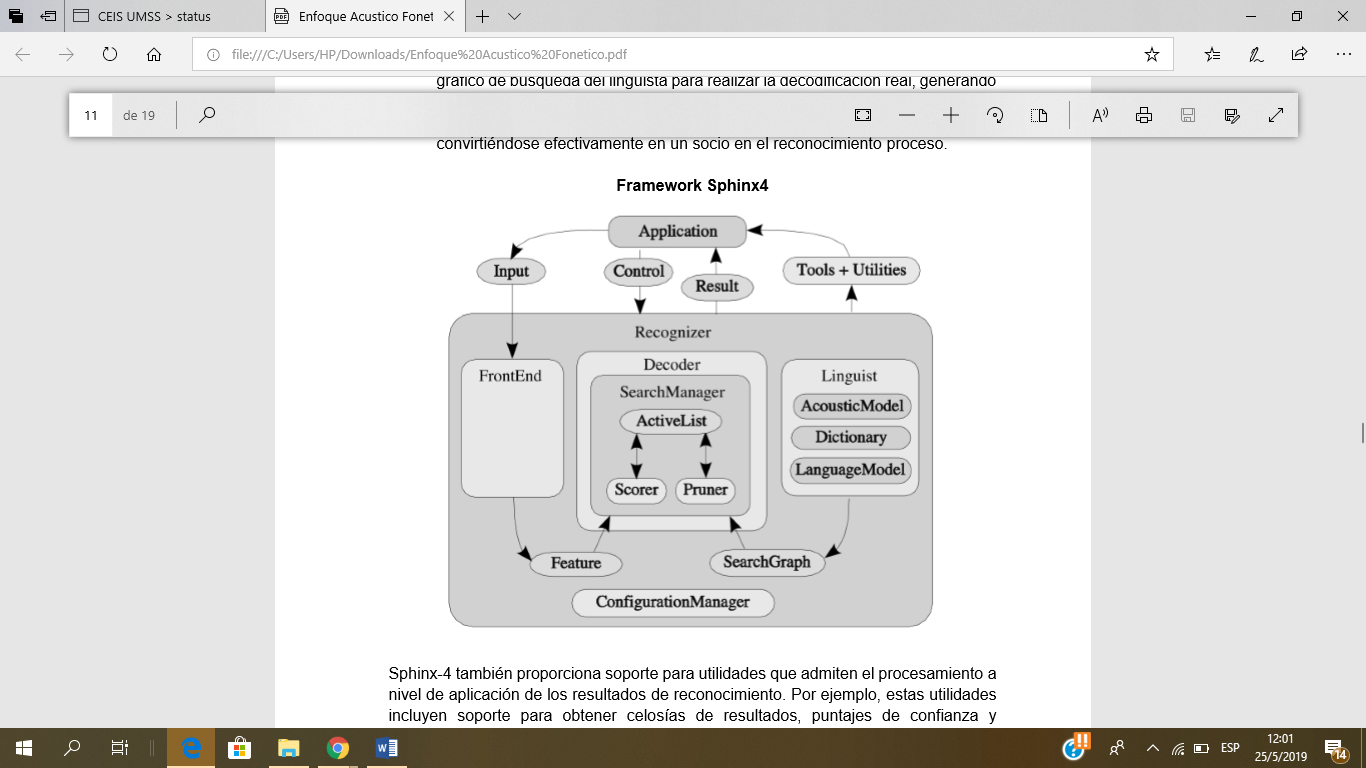
**Algoritmo**

La tecnología de voz establece varios límites importantes a la forma en que implementa una aplicación. Es imposible reconocer cualquier palabra conocida del idioma.

Hay tres módulos principales en el marco Sphinx-4: FrontEnd, Decoder y Linguist.  FrontEnd toma una o más señales de entrada y las parametriza en una secuencia de características.

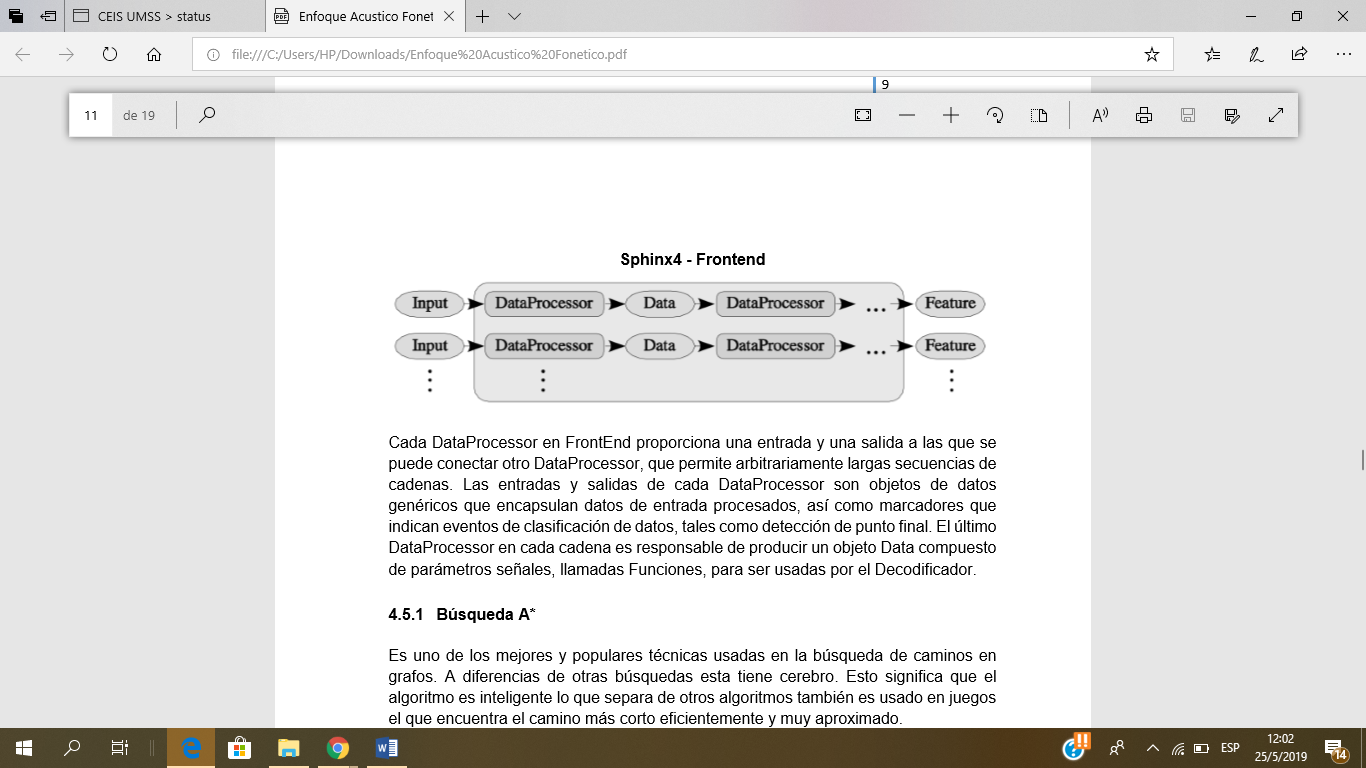
 Linguist traduce cualquier tipo de modelo de lenguaje estándar, junto con información de pronunciación del Diccionario e información estructural de uno o más conjuntos de AcousticModels, en un SearchGraph.

 El SearchManager en el Decoder usa las características del FrontEnd y el gráfico de búsqueda del lingüista para realizar la decodificación real, generando resultados. En cualquier momento antes o durante el proceso de reconocimiento, la aplicación puede emitir Controles a cada uno de los módulos, convirtiéndose efectivamente en un socio en el reconocimiento proceso.



Sphinx-4 también proporciona soporte para utilidades que admiten el procesamiento a nivel de aplicación de los resultados de reconocimiento. Por ejemplo, estas utilidades incluyen soporte para obtener celosías de resultados, puntajes de confianza y comprensión del lenguaje natural.

El objetivo de FrontEnd es parametrizar una señal de entrada (por ejemplo, audio) en una secuencia de características de salida. Como se ilustra en la Figura, el FrontEnd comprende una o más cadenas paralelas de módulos de procesamiento de señal de comunicación reemplazables llamados DataProcessors. El soporte de cadenas múltiples permite el cálculo simultáneo de diferentes tipos de parámetros de la misma o diferentes señales de entrada.



Cada DataProcessor en FrontEnd proporciona una entrada y una salida a las que se puede conectar otro DataProcessor, que permite arbitrariamente largas secuencias de cadenas. Las entradas y salidas de cada DataProcessor son objetos de datos genéricos que encapsulan datos de entrada procesados, así como marcadores que indican eventos de clasificación de datos, tales como detección de punto final. El último DataProcessor en cada cadena es responsable de producir un objeto Data compuesto de parámetros señales, llamadas Funciones, para ser usadas por el Decodificador.