SAUTRELA: UN ENTORNO DE DESARROLLO VERSÁTIL PARA LAS TECNOLOGÍAS DEL HABLA

Mikel Peñagarikano, German Bordel, Sonia Bilbao, Maider Zamalloa, Luis Javier Rodriguez

Grupo de Trabajo en Tecnologías Software
Departmento de Electricidad y Electrónica
Universidad del País Vasco
mikel.penagarikano@ehu.es

RESUMEN

El presente artículo describe Sautrela, un paquete de software desarrollado como código abierto en Java™, altamente modular y escalable, y enfocado a las tecnologías del habla (véase http://sautrela.es). Su arquitectura neutra y modular permite no sólo generar motores de reconocimiento del habla partiendo desde cero, sino también abordar otras tareas relacionadas con las tecnologías del habla, tales como el reconocimiento y verificación de la lengua y del locutor. Los sistemas basados en Sautrela son instanciados a partir de un descriptor XML parametrizable, de tal forma que el valor de algunos de sus parámetros pueda ser establecido en tiempo de ejecución. La arquitectura de modelos por capas y la disponibilidad de módulos de entrenamiento y decodificación de propósito general sirven de base para la construcción de sistemas de complejidad arbitraria.

1. INTRODUCCIÓN

Sautrela es un entorno de desarrollo portable, altamente modular y de código abierto enfocado al procesamiento del habla. Sautrela trata de unificar todas aquellas tareas relacionadas con el reconocimiento del habla, tales como el procesamiento de señal, el modelado (entrenamiento) y la decodificación. Al estar basado en Java™, ofrece una gran capacidad de portabilidad a diversas arquitecturas hardware. Sautrela hace uso de una arquitectura de componentes basada en JavaBeans™ [1], lo que la convierte en una plataforma modular fácilmente ampliable con plugins de terceros.

Si bien en la actualidad existen paquetes de código abierto enfocados al procesamiento del habla, tales como HTK [5] y Sphinx [4], ambos carecen de la flexibilidad necesaria para abordar diferentes tareas que surgen en el ámbito de las tecnologías del habla. Así, HTK es un toolkit que ofrece una amplia gama de herramientas para diseñar y manipular modelos ocultos de Markov (HMM), pero su arquitectura resulta rígida a la hora de integrar otras tecnologías. Sphinx plantea una arquitectura más modular, pero claramente orientada a sistemas de reconocimiento del habla. Sautrela define una arquitectura neutra y modular que permite no solo generar motores de reconocimiento del habla partiendo desde cero, sino también abordar otras tareas relacionadas con las tecnologías del habla, tales como el reconocimiento y verificación de la lengua y del locutor.

El resto del artículo está organizado como sigue. La Sección 2 describe la arquitectura y características principales de Sautrela. La Sección 3 revisa algunos de los módulos de procesamiento incluidos en el paquete. Finalmente, la Sección 4 resume las principales contribuciones del presente trabajo.

2. ARQUITECTURA

La mayoría de las tareas que se abordan dentro de las tecnologías del habla pueden verse como complejos sistemas de procesamiento de la información: un reconocedor del habla no es sino un decodificador que trata de extraer un mensaje que ha sido previamente codificado en una señal sonora, y un verificador de la lengua trata de comprobar si una lengua ha sido o no utilizada en la codificación acústica de un mensaje. Este tipo de sistemas se componen normalmente de fases independientes de procesamiento (eliminación de eco y reverberación, extracción de información espectral, reducción de ruido, decorrelación, compensación de canal, modelado, etc.) cuyo encadenamiento da lugar al procesamiento global. Sautrela se basa en la abstracción de componentes de procesamiento (*Procesadores*) que al encadenarse dan lugar a sistemas complejos de procesamiento (*Engines*).

2.1. Engines y Procesadores

Una *Engine* está compuesta por un conjunto de módulos de procesamiento interconectados entre sí y que son ejecutados en hilos independientes (véase la Figura 1). Dicha interconexión se realiza mediante buffers intermedios donde datos y señales de control son almacenados

Este trabajo ha sido parcialmente financiado por el Gobierno Vasco, dentro del programa SAIOTEK, a través de los proyectos S-PE06UN48 y S-PE07UN43.

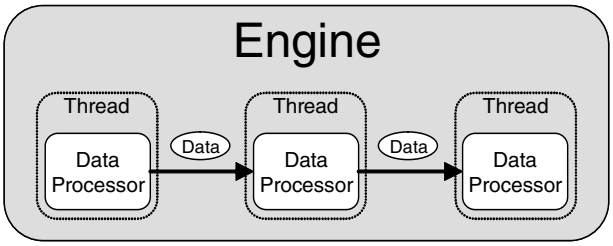

Figura 1. Una *Engine* consta de un conjunto de *Procesadores* encadenados ejecutándose en hilos independientes. Cada módulo de procesamiento representa una fase del proceso global.

temporalmente. La naturaleza multihilo se ajusta perfectamente a las arquitecturas hardware multicore actuales, y permite aprovechar los recursos del sistema. Los *Procesadores* son unidades independientes de procesamiento que implementan una interfaz que les permite ser integrados en el sistema.

Las *Engines* son instanciadas a partir de un simple descriptor XML que contiene la lista de *Procesadores* y los valores de sus parámetros. Es posible también parametrizar una *Engine*, de tal forma que los valores de algunos de sus parámetros puedan ser establecidos en el momento de la ejecución. La parametrización de engines es una técnica sencilla y elegante que permite diseñar sistemas configurables en tiempo de ejecución. La Figura 2 muestra un descriptor XML parametrizado.

2.2. Plugins

Un Plugin es un conjunto de procesadores diseñados para algún tipo de aplicación específica. Basándose en el modelo de componentes software JavaBeans™, Sautrela integra de forma natural tanto plugins propios como plugins desarrollados por terceros. En este sentido, la introspección es una tecnología clave, ya que gracias a ella es posible instanciar y configurar los procesadores contenidos en un plugin, e incluso obtener la documentación necesaria para el usuario final.

2.3. Niveles de ejecución

Sautrela define dos niveles de ejecución: usuario y desarrollador. La información mostrada ante excepciones (errores) dependerá de dicho nivel. Así, un usuario será informado únicamente del error que ha ocurrido, mientras que a un desarrollador (o usuario avanzado) le será mostrada la traza completa de la excepción ocurrida. Esta sencilla distinción de niveles de ejecución ofrece al usuario medio un entorno de ejecución amigable, permitiendo a los desarrolladores acceder a toda la información necesaria para depurar errores.

3. PLUGINS INTEGRADOS

Sautrela viene acompañado de un conjunto de plugins que implementan todas las funcionalidades necesarias para desarrollar un sistema de reconocimiento del habla. A pesar de tratarse de conjuntos diseñados con una finalidad clara, la naturaleza modular de los componentes permite su reutilización para la creación de otro tipo de sistemas, minimizando el esfuerzo requerido. A continuación se describen algunos de estos componentes.

```
<Engine name="Una sencilla Engine">
 <DataProcessor className="class.Name.of.proc1">
  <param name="paramName" value="value" />
 </DataProcessor>
 <Buffer size="3" blocking="true" />
 <DataProcessor className="className.of.proc2">
  <param name="paramName1" value="?-a [23]" />
  <param name="paramName2" value="?-b" />
  <param name="paramName3" value="value3" />
 </DataProcessor>
 <Buffer size="3" blocking="true" />
 <DataProcessor className="className.of.proc3">
  <param name="paramName1" value="value1" />
  <param name="paramName2" value="value2" />
 </DataProcessor>
</Engine>
```

Figura 2. La *Engine* del ejemplo consta de tres *Procesadores*. Los parámetros que no aparecen toman su valor por defecto. El segundo *Procesador* tiene dos parámetros cuyos valores podrán ser establecidos cuando se instancie la engine, mediante las opciones de línea de comando `-a` y `-b`. Si no se presentan dichas opciones, el primer parámetro tomará el valor `23`, y el segundo su valor por defecto.

3.1. Audio

Existe un conjunto mínimo de procesadores para la lectura, captura y reproducción de audio. La utilización de localizadores uniformes de recursos (URL) ofrece una amplia gama de posibilidades de acceso a los recursos de audio, pudiendo estos residir en un simple archivo, un conjunto de archivos comprimidos, la red, etc.

3.2. Base de datos acústica

Sautrela define una estructura de base de datos acústica que permite empaquetar todos los recursos acústicos y sus correspondientes etiquetados (locutor, transcripción ortográfica, transcripción fonética, etc.) mediante un descriptor XML. Cada recurso lleva asociado un identificador único, de tal forma que el módulo de lectura de bases de datos pueda acceder de forma automática a parte de su contenido.

3.3. Procesamiento de señal

El plugin de procesamiento de señal implementa cada una de las fases de la parametrización de la señal de voz: preénfasis, ventaneo, discriminación voz/no-voz, FFT, banco de filtros, DCT, normalización cepstral, Feature-Warping y obtención de derivadas. Además, se cuenta con herramientas de clustering y un módulo de cuantificación

vectorial. Todos estos componentes son parametrizables, y sus valores por defecto están optimizados para recursos de audio grabados en condiciones de laboratorio.

3.4. Modelado, entrenamiento y decodificación

Sautrela cuenta con un conjunto de modelos que da cobertura al modelado acústico, léxico y del lenguaje. Básicamente consta de modelos ocultos de Markov discretos y continuos, autómatas de estados finitos deterministas y no deterministas, y modelos K-explorables en sentido estricto (n-gramas). También incluye los modelos de Markov por capas (Layered Markov Models), los cuales permiten integrar en un solo autómata/modelo diversos niveles de conocimiento [2]. Los modelos por capas son una pieza clave de la arquitectura de modelos de Sautrela, ya que gracias a ellos es posible aplicar diferentes técnicas de aprendizaje y decodificación.

La arquitectura de modelos de Sautrela es la base sobre la cual se asientan dos procesadores fundamentales: el decodificador y el entrenador. Ambos pueden funcionar frente a cualquiera de los modelos previamente citados. Dichos modelos implementan una interfaz de decodificación que, en combinación con parámetros de poda, es utilizada para obtener la decodificación más probable dada una entrada de datos/audio. De esta forma, el mismo modulo sirve tanto para crear un sencillo decodificador acústico-fonético, un reconocedor de comandos, o un complejo reconocedor de habla continua, ya que la única diferencia reside en el modelo que utilicemos. De forma análoga, los modelos implementan una interfaz de entrenamiento que permite desacoplar la optimización de la interfaz (tarea realizada por el entrenador) del ajuste interno de parámetros (tarea realizada por el modelo) [3]. Un único módulo de entrenamiento se encarga de realizar las tareas de reestimación independientemente de cual sea la naturaleza interna del modelo en cuestión.

Cabe mencionar que la arquitectura de modelos de Sautrela es ampliable: todo modelo que implemente la interfaz de decodificación-entrenamiento es directamente integrable en el sistema.

3.5. Utilidades

Existe un conjunto de componentes genéricos que ofrecen ciertas utilidades, como la monitorización, volcado y recuperación de datos. El volcado y recuperación de datos resulta muy útil en situaciones en las que una misma *Engine* que contiene una sección inicial constante es ejecutada repetidamente. Por ejemplo, al entrenar modelos acústicos es posible realizar una sola vez la parametrización del conjunto de entrenamiento de la base de datos y volcar el flujo de datos resultante. Una vez volcado, ese flujo de datos puede ser usado como entrada de la fase de entrenamiento de los modelos, procedimiento que se repetirá tantas veces como sea preciso.

4. CONCLUSIONES

En este artículo se ha presentado Sautrela, un entorno de desarrollo modular, escalable y de código abierto enfocado a las tecnologías del habla. Se trata de un software de propósito general que cuenta con un conjunto de plugins diseñados para construir sistemas de reconocimiento del habla. En la actualidad, Sautrela está siendo utilizado también en las áreas de verificación de la lengua y del locutor, ya que sirve de base tecnológica para las distintas tareas abordadas por el Grupo de Trabajo en Tecnologías Software (GTTS).

5. BIBLIOGRAFÍA

[1] G. Hamilton. The JavaBeans^{MT} API specification. Technical report, Sun Microsystems, 1997.

[2] M. Penagarikano and G. Bordel. Layered Markov Models: A New Architectural Approach to Automatic Speech Recognition. In *Proceedings of the MLSP Workshop*, pages 305–314, Sao Luis, Brasil, Octover 2004.

[3] M. Penagarikano, G. Bordel, and L. J. Rodriguez. Unified training of WFSA through a generic interface. In *Proceedings of Spoken Language Technology Workshop, 2006. IEEE*, pages 122–125, December 2006.

[4] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. Sphinx-4: A flexible open source framework for speech recognition. Technical Report TR-2004-139, Sun Microsystems, 2004.

[5] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4*. Cambridge University Engineering Department, Cambridge, UK, 2006.