SPEAKIT! V3.0

MANUAL DE USUARIO

Estructura general del sistema

Speakit es un sistema que lee documentos por salida de audio y los guarda indexados y comprimidos.

El sistema Speakit, permite ingresar un documento de texto, grabar todas las palabras que no estén registradas, y agregarlas a un diccionario que asocia palabras con su audio, y también permite reproducir las palabras contenidas en un documento.

Contiene un módulo de indexación y almacenamiento de documentos, que en esta versión tiene la posibilidad (opcional) de guardar los documentos en forma comprimida soportando

2 métodos de compresión distintos (PPMC y LZP). Además contiene un módulo stand alone que permite comprimir archivos unicode codificados en UTF-16BE con el método aritmético dinámico.

El módulo Speakit, tiene la función de agregar el audio de una palabra. Además tiene la función de agregar un documento a la colección de documentos almacenados. Esta función realiza su tarea valiéndose del módulo documents. Puede almacenar documentos a pesar de que el módulo dictionary no contenga todas sus palabras.

Audiofile tiene implementación secuencial y está indizado por un Trie. El trie fue implementado sobre un archivo directo de registros por bloque que almacena los nodos del árbol y un archivo secuencial que se utiliza como índice para saber en que bloque buscar un nodo sabiendo su número. El archivo directo está implementado independientemente en la clase DirectRecordFile para poder ser reutilizado por otras estructuras que requieran el uso de un archivo directo.

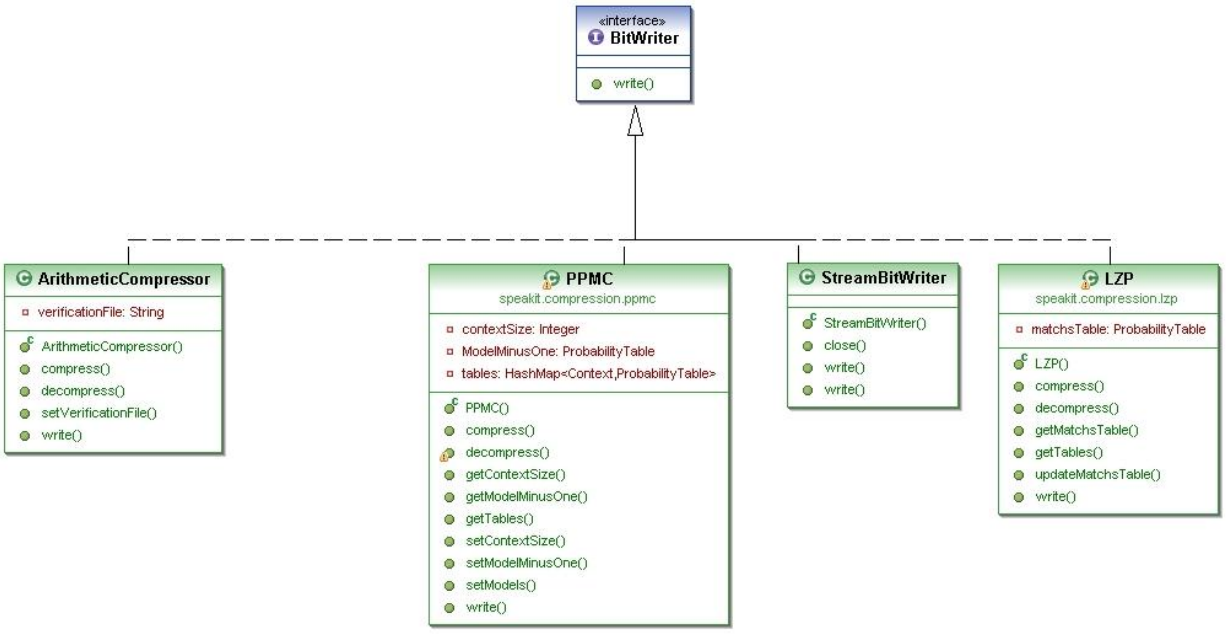
El módulo *documentsstorage* tiene la funcionalidad de guardar, y buscar los documentos de texto guardados conociendo el offset dentro del archivo. Cuando se agrega un nuevo documento el módulo ftrs extrae los terminos relevantes, dejando de lado los q se repiten comunmente, tales como artículos, proposiciones, etc. Si se agregan al sistema un conjunto de documentos estos se indexan de una manera más óptima que ingresándolos de a uno, creando las listas invertidas sincronizadamente y utilizando un algoritmo de sort externo. Los documentos agregados, serán comprimidos en la próxima etapa con el módulo compression.

La búsqueda de documentos se realiza a través del módulo *ftrs* y se recuperan a través del módulo *documentsstorage*. Las listas invertidas del ftrs se almacenan en un archivo directo de registros por bloque que está indexado por un árbol B# para agilizar las búsquedas. El árbol B# está implementado en un módulo independiente y en el módulo ftrs se lo utiliza junto con un encoder que permite comprimir los términos en las hojas mediante front coding.

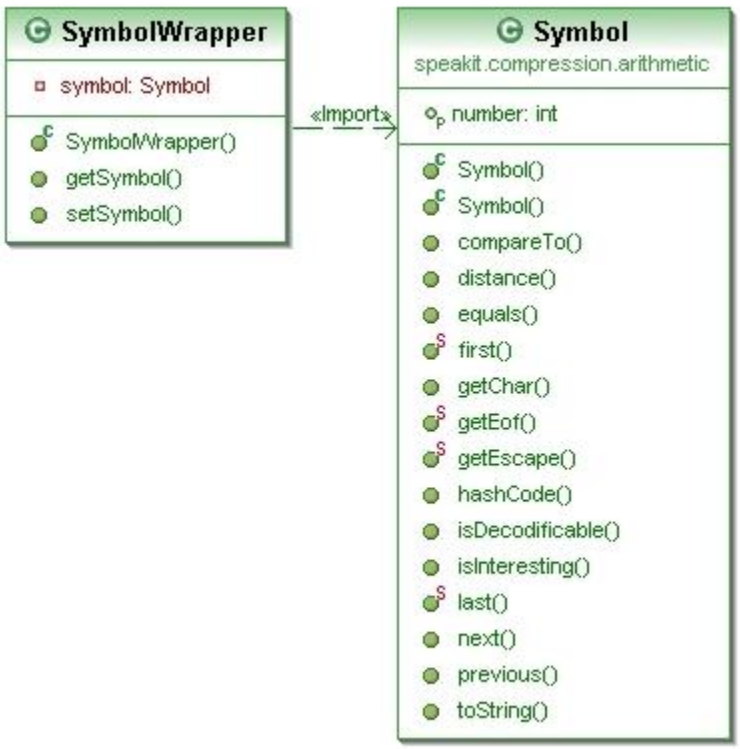
La garantía de funcionalidad y estabilidad del sistema está soportada por unas 300 pruebas automáticas, las cuales incluyen pruebas simples y pruebas de stress. Las pruebas de stress utilizadas garantizan por ejemplo que el árbol B# pueda indexar todas las palabras del idioma castellano (véase StressTreeTest con speakit/test/files/lemario.txt), u otras por ejemplo garantizan la robustez del sistema ftrs, y en esta versión hay pruebas que garantizan que los datos comprimidos se puedan recuperar. Estas pruebas se pueden utilizar a modo de documentación técnica adicional para aprender sobre el funcionamiento y la utilización del sistema, por lo cual recomendamos su lectura y ejecución.

Compresión Aritmética

En esta etapa se implementó un compresor aritmético dinámico, de orden 0, capaz de comprimir archivos de texto en formato unicode. Como el unicode tiene 65536 caracteres posibles hubo que utilizara precisión de 32 bits para la representación el piso y el techo del intervalo.



Symbol



Encapsula a un caracter unicode.

ProbabilityTable

Mantiene la lista de símbolos con su respectiva probabilidad. Permite notificar que un símbolo ha ocurido para que incremente su probabilidad, o bien agregarlo a la lista si no habia ocurrido.
Además tiene dos funcionalidades cruciales para la codificación y decodificación aritmética:
1_ getSymbolFor:
Este método se utiliza para la decodificación, permite obtener un símbolo a partir del valor de la ventana de inspeccion y del Rango actual. Para lograrlo asigna a cada símbolo de la lista un subintervalo (piso y techo) dentro del rango actual, con tamaño del subintervalo proporcional a la magnitud de su probabilidad. Luego busca dentro de cual de estos intervalos está el número de la ventana de inspección y devuelve el símbolo asociado a ese subintervalo.

2_ zoomIn:
Este método se utiliza en la codificación, sirve para hacer zoom en el rango para un símbolo determinado. Esto lo hace calculando el subintervalo correspondiente al símbolo indicado de la misma forma que se hace en el metodo getSymbolFor.

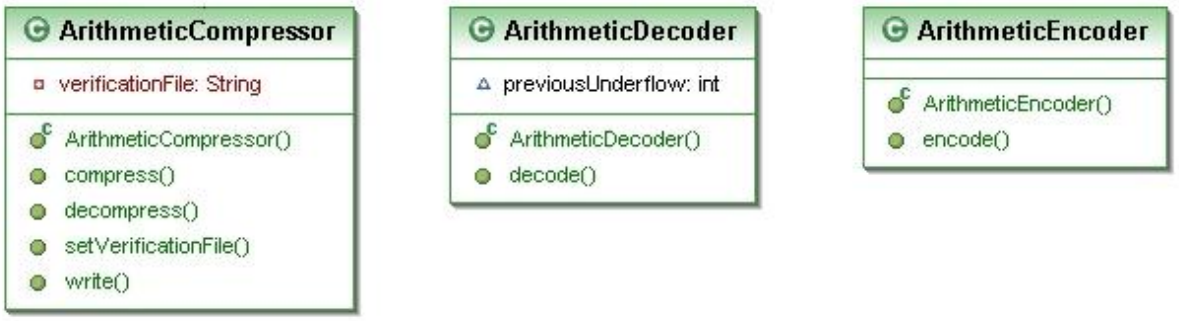
ArithmeticEncoder

Esta clase se encarga de codificar un símbolo en bits comprimidos que son emitidos a traves de un BitWriter. Recibe como parámetro una tabla de probabilidades. Utilizado en los otros compresores y en ArithmeticCompressor.

ArithmeticDecoder

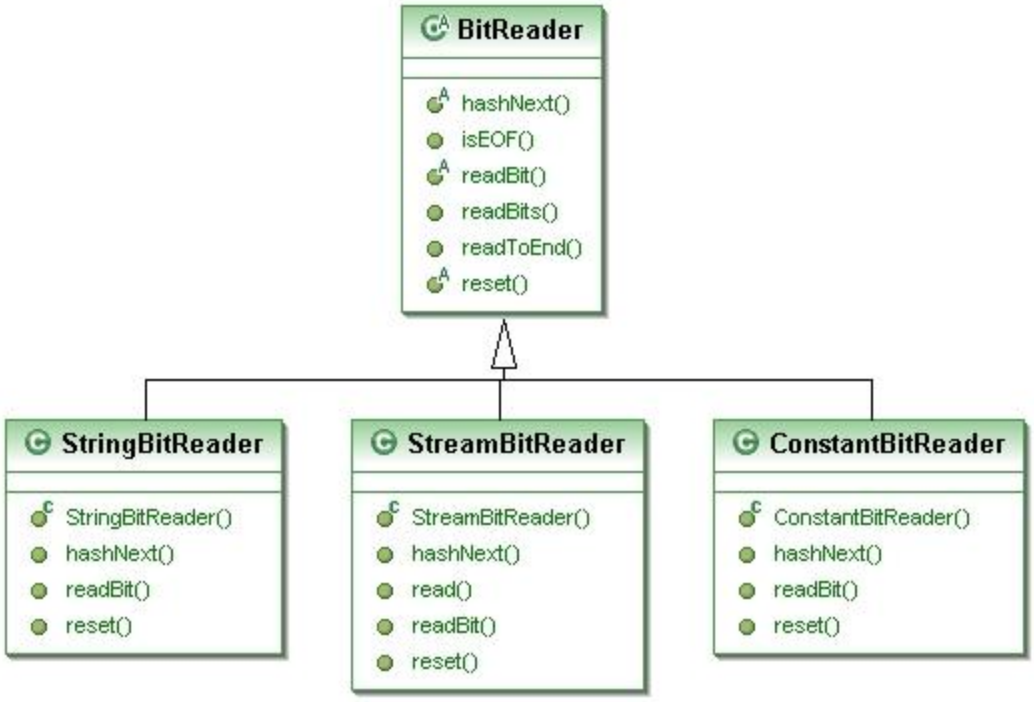
Se encarga de recuperar símbolos del archivo comprimido y mover la ventana de inspección. Utilizado en los otros compresores y en ArithmeticCompressor.

ArithmeticCompressor



Compresor aritmético dinámico. Utiliza el ArithmeticEncoder y ArithmeticDecoder para comprimir y descomprimir un archivo respectivamente. Mantiene tablas de probabilidades de los símbolos que van apareciendo en el archivo mientras de codifica/decodifica.
Este es el módulo que se utiliza para comprimir desde el menú de speakit.

StreamBitReader

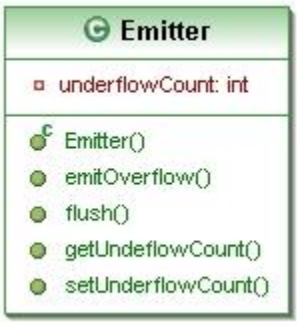


Esta clase sirve para desempaquetar bytes de un stream y convertirlos en bits.

StreamBitWriter

Esta clase sirve para empaquetar bits en bytes para luego escribirlos a un stream.

Emitter



Encapsula el comportamiento de la emisión de bits del aritmético. Tiene primitivas para emitir overflow e incrementar el contador de underflow. Las emisiones se van acumulando

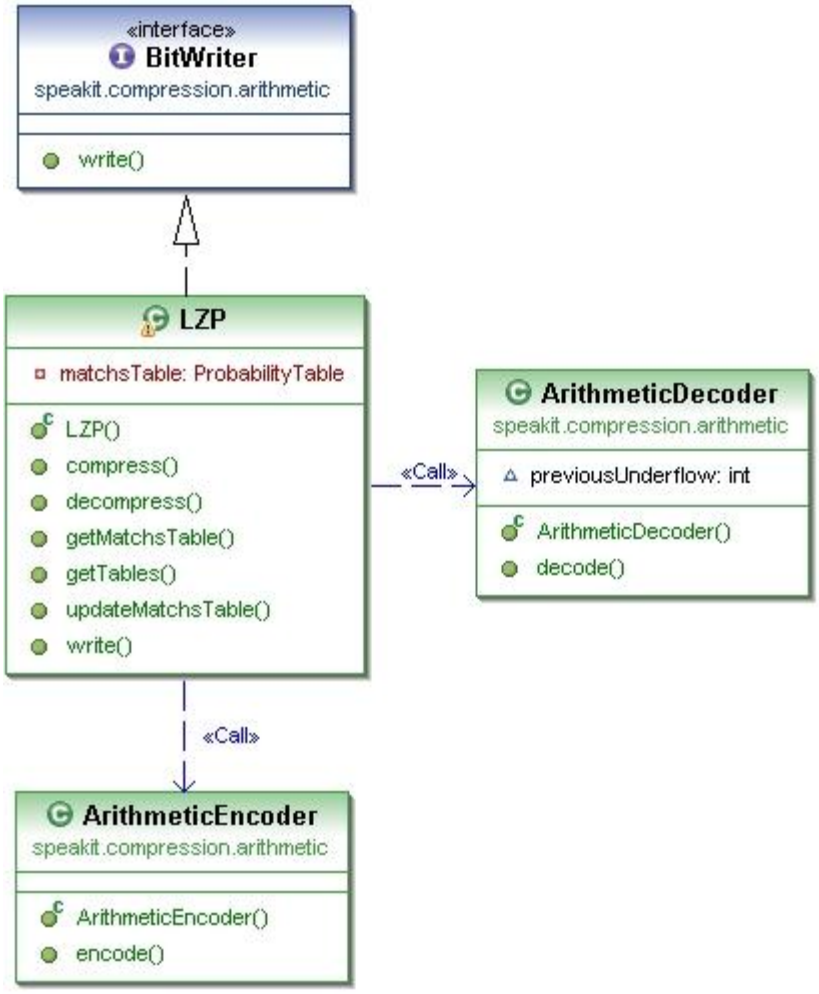
dentro de este objeto hasta que se ejecuta el metodo flush que devuelve los bits en forma de String.

Range

Esta clase encapsula el comportamiento del intervalo del compresor aritmético, mantiene los valores de piso y techo y provee metodos para interactuar con ellos. Tiene un método para hacer zoom en un subintervalo, actualiza los correspondientes extremos y resuelve el overflow y el underflow emitiendo los bits que correspondan mediante un Emitter.

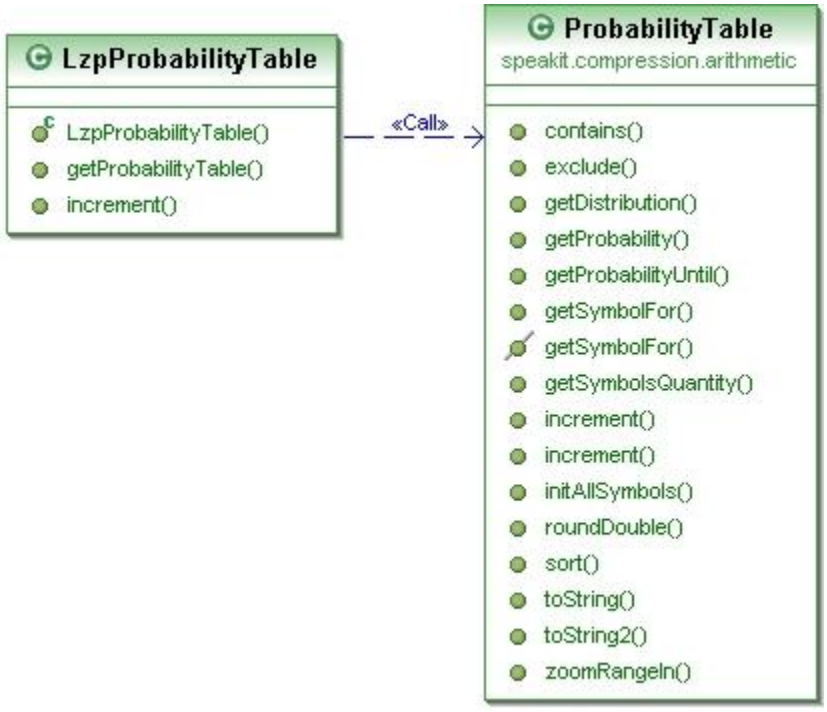
Compresión LZP

LZP



Clase encargada de modelar el compresor LZP.

LzpProbabilityTable



Modelado de las tablas de probabilidad, segun la necesidad del algoritmo de compresión LZP. La misma mantiene los caracteres con frecuencia de aparición distinta de 0 (distinta de 1 en la implementacion) para cada contexto y construye las ProbabilityTable según sean solicitadas.

LZPTable

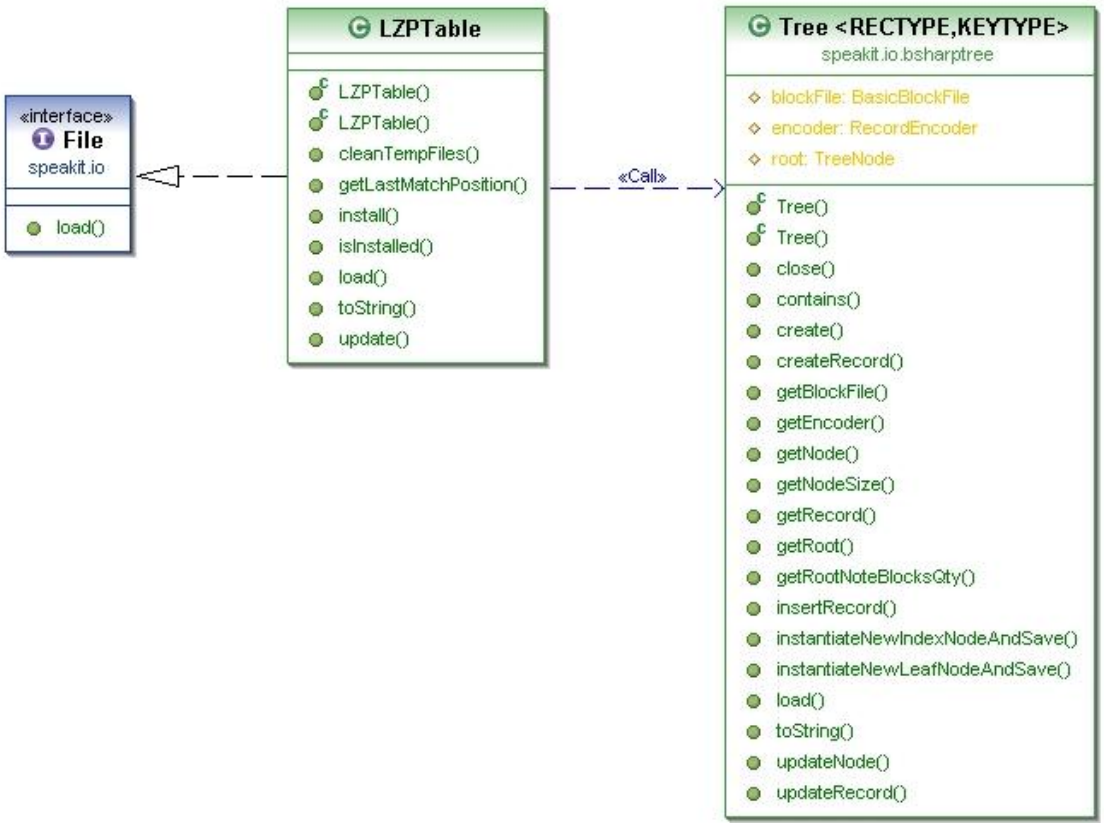
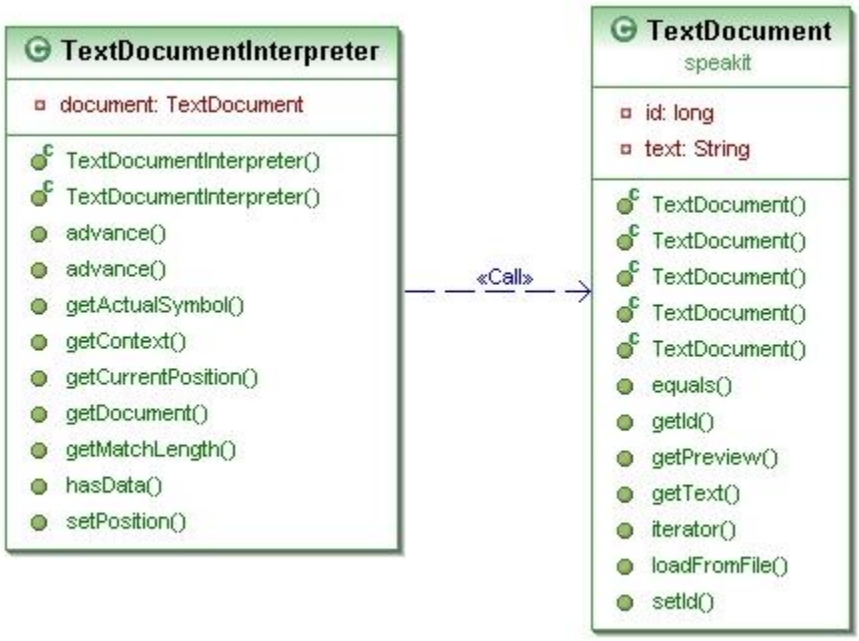


Tabla de trabajo del LZP, registra en cada paso la posición del contexto de busqueda de matchs. La misma se almacena en disco, no superando nunca el tamaño de 4096 bytes de memoria. El almacenamiento se realiza en una estructura de arbol B# para que la busqueda sea en el menor tiempo posible y asi acortar los tiempos de compresion y descompresion.

LZPTableRecord

Representacion de los registros que se insertaran en el árbol B#, los mismo almacenan el contexto y la última aparición del mismo dentro del documento que se está comprimiendo.

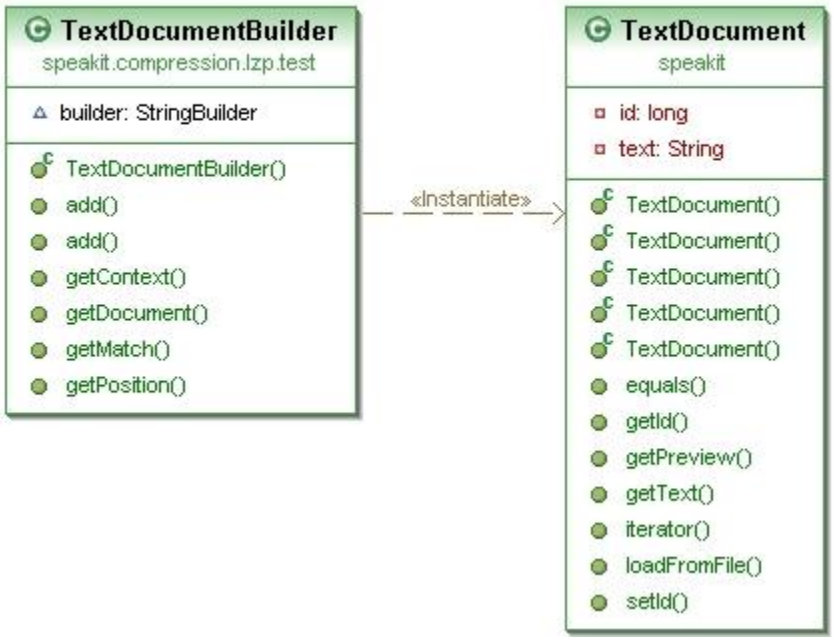
TextDocumentInterpreter



Esta clase se encarga de encapsular la forma de recorrer un documento, proveyendo en cada paso la posición actual, el contexto en el que se esta trabajando y la posibilidad de

referenciar el texto desde cualquier posicion del mismo.

TextDocumentBuilder



Clase que facilita el armado de un documento y provee datos sobre el mismo, como ser contexto actual, posición y permite recorrer los caracteres ya decodificados.

Algunas consideraciones

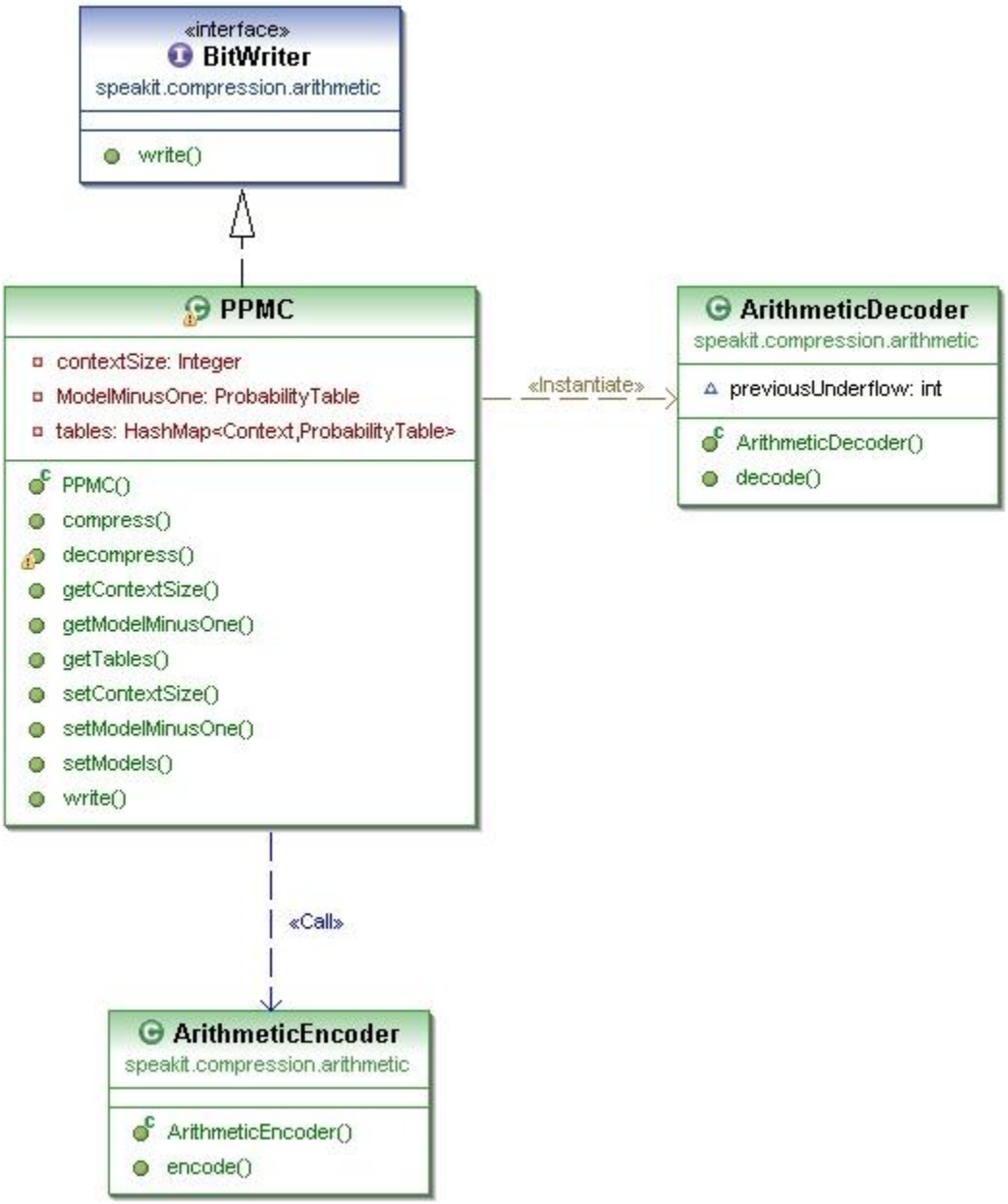
Una de las limitaciones que presentaba el algoritmo estaba en el tamaño de la tabla de trabajo, en este modelo llamada LzpTable, la misma no podia superar los 4096 bytes de memoria. Para atacar este problema, se decidió, como primera medida, hacer una actualizacion optima de la misma, evitando tener contextos repetidos, ya que los únicos que tienen relevancia son los que se han registrado al final.
Por otro lado, también se implemento una estructura de árbol BSharp que permite mantener un solo nodo del mismo en memoria, cumpliendo de esta manera la restricción de tamaño y premitiendo la busqueda y la actualización de una manera rápida.

Compresión PPMC

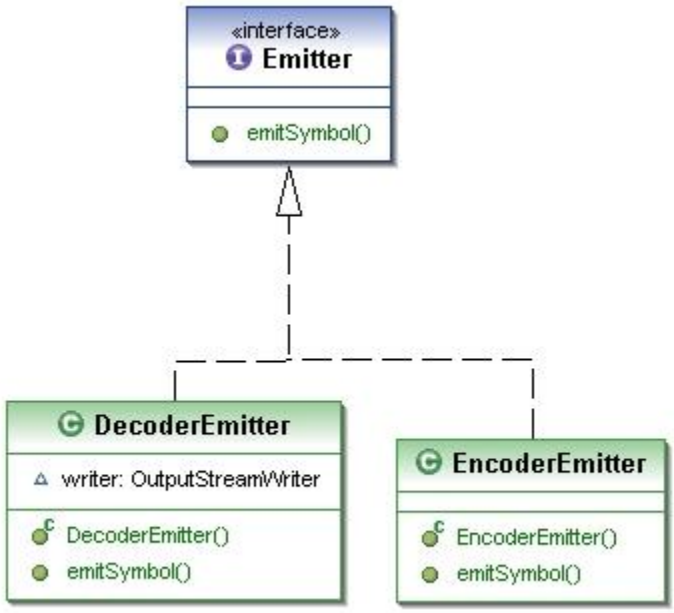
En esta parte del TP tuvimos que implementar un compresor PPMC con contextos variables entre 0 y 4. Esta estructura tiene los métodos **compress**, al que se le debe pasar un **TextDocument** y que utiliza la clase **ArithmeticEncoder** para generar la cadena de

bits correspondiente y **decompress** a los que se les debe pasar una cadena de bits y utiliza el **ArithmeticDecoder** para descomprimirla. Para llevar la estructura de los modelos se decidió hacer un hashmap de **Context** y **ProbabilityTable**. Además, el modelo -1 como tiene un tratamiento especial en muchas ocasiones (no se incrementa la probabilidad de un símbolo al emitirlo, no tiene caracter de escape, etc.) se decidió tenerlo apartado del hashmap anteriormente mencionado.

PPMC



Emitter



Para realizar la emisión, ya sea de bits al comprimir o de caracteres al descomprimir, se utiliza una interfaz Emitter con un método emitSymbol() que será implementado en el **EncoderEmitter** y en el **DecoderEmitter**

EncoderEmitter

En esta clase se implementa la interfaz **Emitter** y emite una cadena de bits, segun un simbolo pasado.

DecoderEmitter

En esta clase se implementa la interfaz **Emitter** y emite un simbolo, segun una cadena de bits pasada.

Instrucciones de uso

Instalación del sistema

Para instalar y correr el sistema se requiere de la herramienta **Apache Ant**. Esta herramienta puede conseguirse mediante el gestor de paquetes de su distribución de Linux o descargandolo de la página **http://ant.apache.org** . Si se elige esta última opción se deben seguir las instrucciones de instalación de la herramienta proporcionadas en la página.

Para compilar o correr la aplicación, utilizando la consola del sistema operativo, ingresar al directorio **speakit** y ejecutar alguno de los siguientes comandos:

ant clean - Usando este comando se borran las carpetas utilizadas para compilar y distribuir la aplicación.

ant compile - Con este comando se compila la aplicación, generando los archivos .class dentro de la carpeta **build** del proyecto.

ant run - Al ejecutar este comando se compila y se ejecuta la aplicación.

ant - Si se ejecuta el comando ant, sin utilizar ningún parametro, por default se ejecuta el comando **ant run**.

Configuración del sistema

El archivo de configuración se llama Speakit.conf y vá ubicado en la raiz de la aplicación. Es un archivo de texto donde cada linea define una variable distinta del sistema.

Linea 1: Tamaño de nodo del arbol bsharp del indice invertido. Este numero va expresado en bytes.
Linea 2: Profundidad del trie.
Linea 3: Indica si al indexar documentos se van a omitir ingresar las palabrasdesconocidas. Valores posibles: 1 (pedir palabras desconocidas), 0 (no pedir palabras desconocidas)
Linea 4: Tamaño de contexto del compresor PPMC. Valores posibles 0,1,2,3,4.

Ejemplo de archivo de configuracion:
1024
4
0
2

Encoding de documentos

En esta versión Speakit **sólo** trabaja con archivos de texto Unicode, formateados en UTF-16BE (Big Endian), tanto para indexar como para reproducir y comprmir. Si se quieren ingresar archivos con otra codificación no garantizamos que funcione correctamente.

Utilización del sistema

Al iniciar la aplicación podemos ver la siguiente pantalla:

```
Menu Principal
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta


0.- Salir
```

Aqui tenemos 6 opciones:

- **Procesar archivo de Texto**: En este módulo el sistema pregunta por un archivo de texto para ser leido. Si el archivo está dentro de la carpeta de la aplicación no es necesario ingresar la ruta al mismo, solo se debe ingresar el nombre. Una vez ingresado el nombre del archivo, el sistema reconoce las palabras que ya están agregadas al diccionario y las que no se encuentren en este, se requerirá que sean grabadas por el usuario. Luego de grabar cada palabra el sistema reproducirá la palabra, preguntará al usuario si la palabra se grabó correctamente y dará la posibilidad de regrabarla si asi lo desea el usuario.

```
Speak It!
Menu Principal
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta
5.- Comprimir un archivo con Aritmético dinámico

0.- Salir
1
Leer archivo de Texto

Ingrese la ruta a continuación:
(Si su archivo es '1.txt' sólo presione ENTER)
hamlet.txt
El documento contiene palabras desconocidas, que deberá grabar a
continuación.

Palabra 'ser'. (ENTER para grabar).

Grabando... (ENTER para detener).

Reproduciendo...(ENTER para confirmar. N para volver a grabar)

Palabra 'o'. (ENTER para grabar).
```

```
Grabando... (ENTER para detener).

Reproduciendo...(ENTER para confirmar. N para volver a grabar)

Palabra 'no'. (ENTER para grabar).

Grabando... (ENTER para detener).

Reproduciendo...(ENTER para confirmar. N para volver a grabar)

Palabra 'esta'. (ENTER para grabar).

Grabando... (ENTER para detener).

Reproduciendo...(ENTER para confirmar. N para volver a grabar)
n
Palabra 'esta'. (ENTER para grabar).

Grabando... (ENTER para detener).

Reproduciendo...(ENTER para confirmar. N para volver a grabar)

Palabra 'es'. (ENTER para grabar).

Grabando... (ENTER para detener).

Reproduciendo...(ENTER para confirmar. N para volver a grabar)

Palabra 'la'. (ENTER para grabar).

Grabando... (ENTER para detener).

Reproduciendo...(ENTER para confirmar. N para volver a grabar)

Palabra 'cuestion'. (ENTER para grabar).

Grabando... (ENTER para detener).

Reproduciendo...(ENTER para confirmar. N para volver a grabar)

El documento fué agregado con éxito.
```

Si se ingresa un nombre de archivo que el sistema no puede encontrar, se emitirá un mensaje de error y la aplicación volvera al menu inicial.

```
Speak It!
Menu Principal
```

```
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta
5.- Comprimir un archivo con Aritmético dinámico

0.- Salir
1
Leer archivo de Texto

Ingrese la ruta a continuación:
(Si su archivo es '1.txt' sólo presione ENTER)
lalala.txt
No pudo encontrarse el archivo 'lalala.txt'.
Speak It!
Menu Principal
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta
5.- Comprimir un archivo con Aritmético dinámico

0.- Salir
```

- Procesar varios archivos de Texto: Funciona igual a la opción anterior, sólo que permite ingresar mas de un documento a la vez. Los nombres de archivo se deben escribir uno a continuación del otro separándolos por una coma.

```
Menu Principal
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta
5.- Comprimir un archivo con Aritmético dinámico

0.- Salir
2
Ingrese cada una de las rutas de los documentos que desea ingresar
separadas por coma
hamlet.txt,speakit.txt

Los documentos ingresados contienen palabras desconocidas que deberá
grabar a continuación
Palabra 'una'. (ENTER para grabar).
```

- **Reproducir Archivo**: Con esta opción podemos reproducir las palabras grabadas con

anterioridad en la aplicación. Se solicitará nuevamente el nombre del archivo ingresado y el sistema indicará por pantalla las palabras a reproducir consecutivamente y se escuchará su audio a continuación.

```
Speak It!
Menu Principal
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta
5.- Comprimir un archivo con Aritmético dinámico

0.- Salir
3
Ingrese la ruta a continuación:
(Si su archivo es '1.txt' sólo presione ENTER)
hamlet.txt
Se va a reproducir el siguiente documento
Ser o no ser: ésta es la cuestión!
```

- Realizar una consulta: Escriba los términos por los que quiere buscar y el sistema desplegará un listado de los documentos más relevantes.

```
Speak It!

Menu Principal
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta
5.- Comprimir un archivo con Aritmético dinámico

0.- Salir
4
Ingrese la consulta
cuestion
Los documentos encontrados para la consulta realizada se muestran a
continuacion:
1 : Ser o no ser: ésta es la cuestión!....
```

Si quiere reproducir uno de los documentos listados, presione 1 y a continuación escriba el número de documento que quiere escuchar y el sistema se lo leerá.

```
Si desea reproducir algun documento presione 1
Para realizar una nueva consulta presione 2
Para ir al menu principal presione 0
```

```
1
Elija el numero de documento que desea reproducir
1
Se va a reproducir el siguiente documento
Ser o no ser: ésta es la cuestión!
```

- **Comprimir un archivo con Aritmético dinámico**: Con esta opción podemos comprimir un documento utilizando el compresor aritmético dinámico. Se solicitará el nombre de un archivo de texto y el sistema lo comprimirá mostrando por pantalla los información sobre el proceso. Speakit creará un archivo comprimido concatenandole al nombre la extensión .zipit. Luego calculará la tasa de compresión. Adicionalmente speakit descomprimirá el archivo recien comprimido generando un nuevo archivo con extensión .zipit.dezipit para compararlo contra el archivo original.

```
Speak It!
Menu Principal
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta
5.- Comprimir un archivo con Aritmético dinámico

0.- Salir
5
Ingrese la ruta a continuación:
(Si su archivo es '1.txt' sólo presione ENTER)
articulo.txt
Comprimiendo...
Arhivo comprimido: articulo.txt.zipit
Tasa de compresión: 36%
Descomprimiendo...
Verificando...
Verificación exitosa.
```

- **Salir**: Esta opción permite salir de **SpeakIt**.

```
Menu Principal
1.- Procesar un archivo de Texto
2.- Procesar varios archivos de Texto
3.- Reproducir Archivo
4.- Realizar una consulta
5.- Comprimir un archivo con Aritmético dinámico

0.- Salir
0
Terminado.
```

Ejemplos

Se documentan las siguientes pruebas hechas con el compresor aritmético con archivos de diferentes tamaños. Se proveen archivos originales codificados en UTF-16BE para poder reproducir las pruebas.

Test Case 1

Ingrese la ruta a continuación:
victima-influenza.txt
Comprimiendo...
Arhivo comprimido: victima-influenza.txt.zipit
Tasa de compresión: 27%
Descomprimiendo...
Verificando...
Verificación exitosa.

Test Case 2

Ingrese la ruta a continuación:
arquitectura-speakit.txt
Comprimiendo...
Arhivo comprimido: arquitectura-speakit.txt.zipit
Tasa de compresión: 37%
Descomprimiendo...
Verificando...
Verificación exitosa.

Test Case 3

Ingrese la ruta a continuación:
airbus.txt
Comprimiendo...
Arhivo comprimido: airbus.txt.zipit
Tasa de compresión: 39%
Descomprimiendo...
Verificando...
Verificación exitosa.

Test Case 4

Ingrese la ruta a continuación:
hola-mundo.txt
Comprimiendo...
Arhivo comprimido: hola-mundo.txt.zipit
Tasa de compresión: -13%
Descomprimiendo...
Verificando...
Verificación exitosa.

Test Case 5

Ingrese la ruta a continuación:
constitucion.txt
Comprimiendo...
Arhivo comprimido: constitucion.txt.zipit
Tasa de compresión: 61%
Descomprimiendo...
Verificando...
Verificación exitosa.