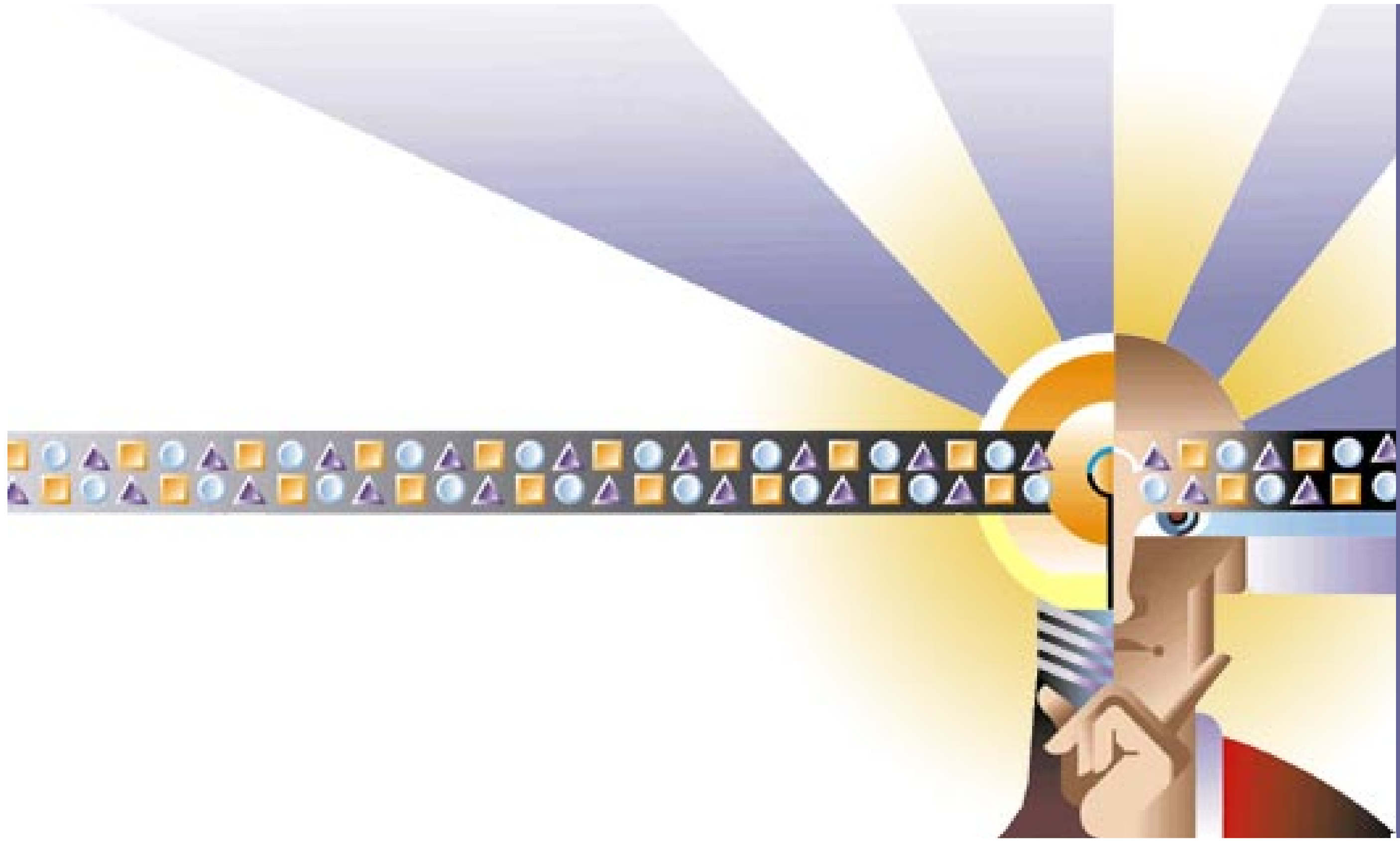


Sphinx 4

Code Walk-Through

Sphinx 4 Team – February 6, 2003

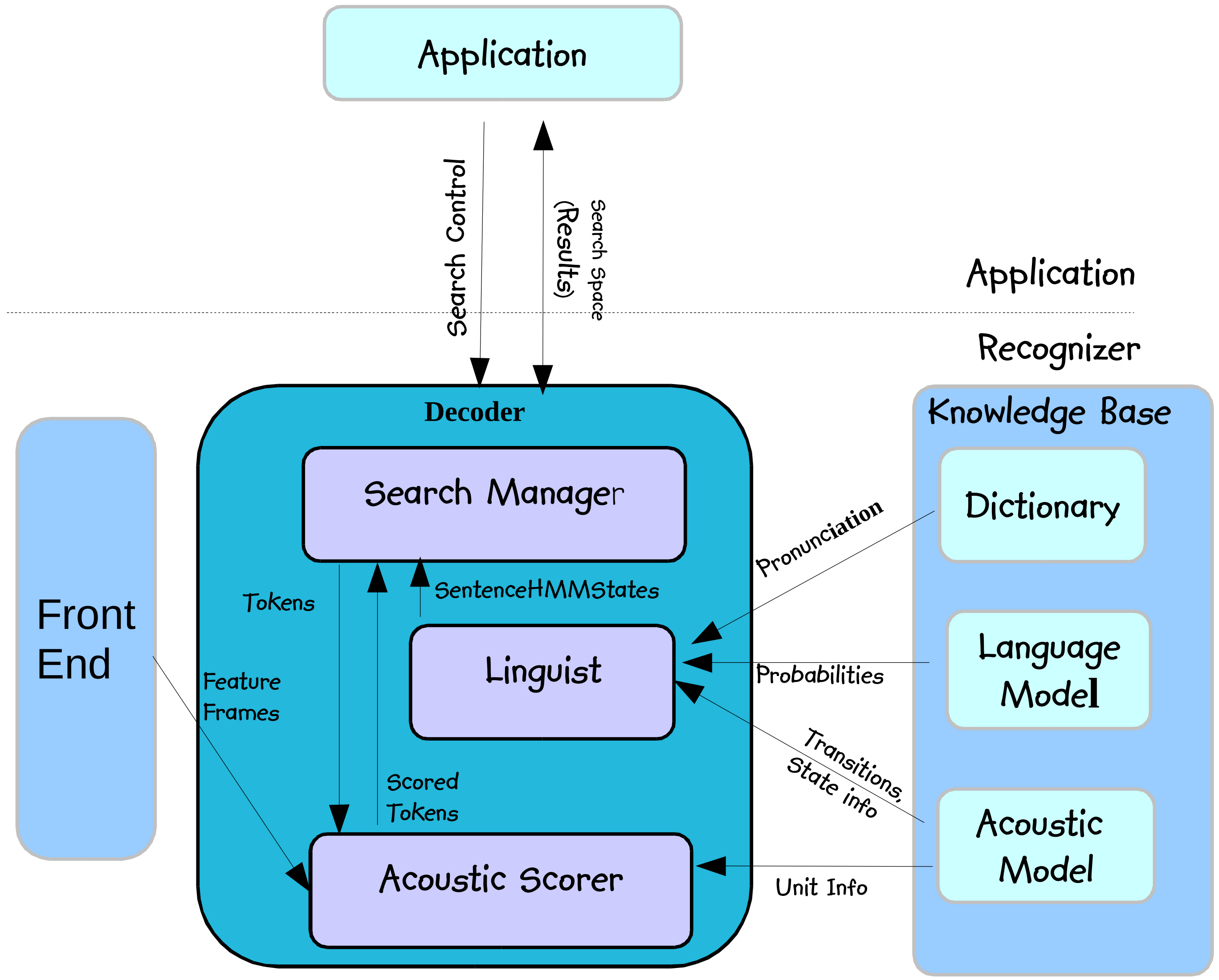
Agenda

- Introduction
- Architecture Overview
- Decoder Walkthrough
- Front End Walkthrough
- Knowledge Base Walkthrough
- Tools and Utilities
- Application

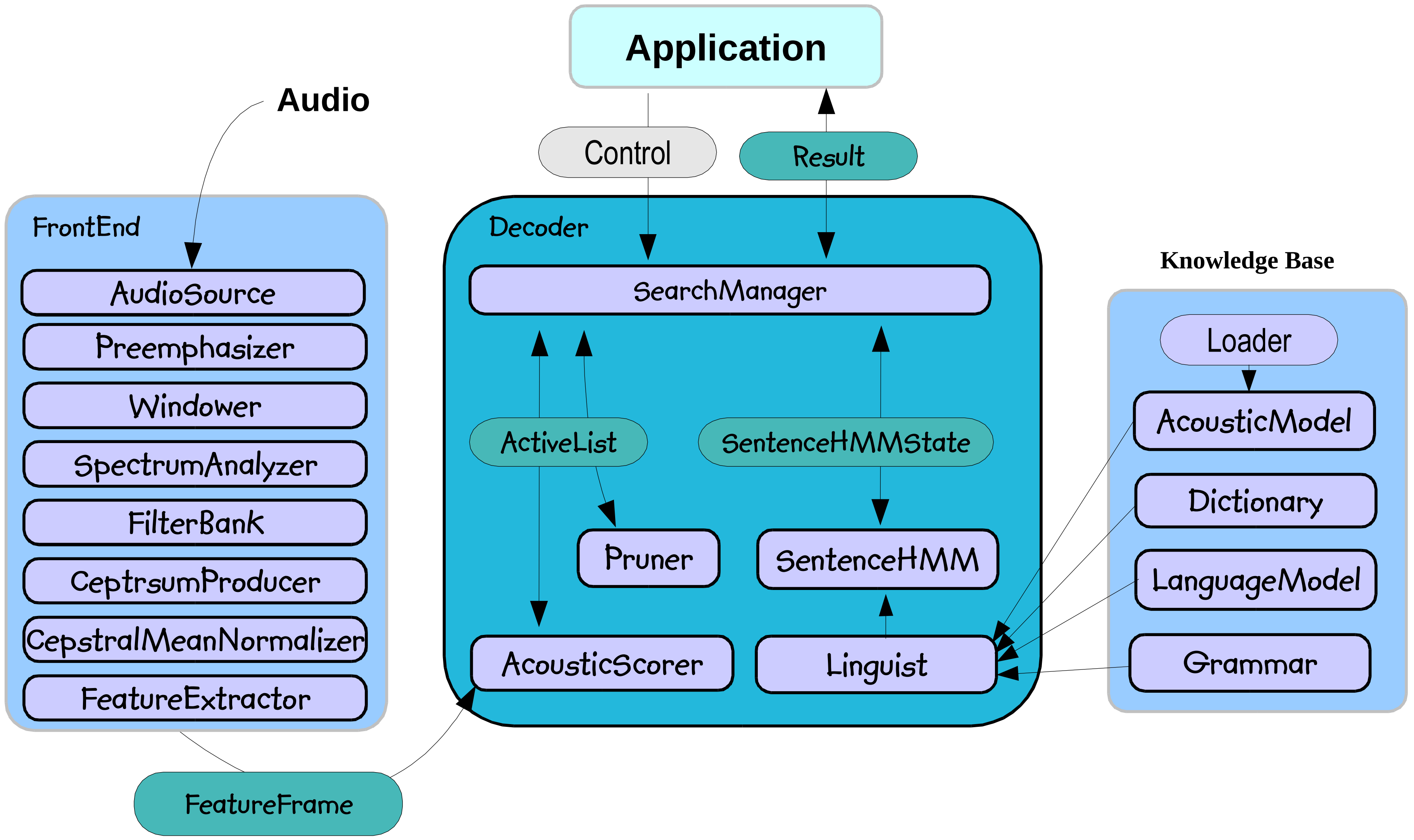
Introduction

- **Goal** – give people a working knowledge of the S4 code
- Present the major classes and interfaces
- Not a design review
- Not a code review

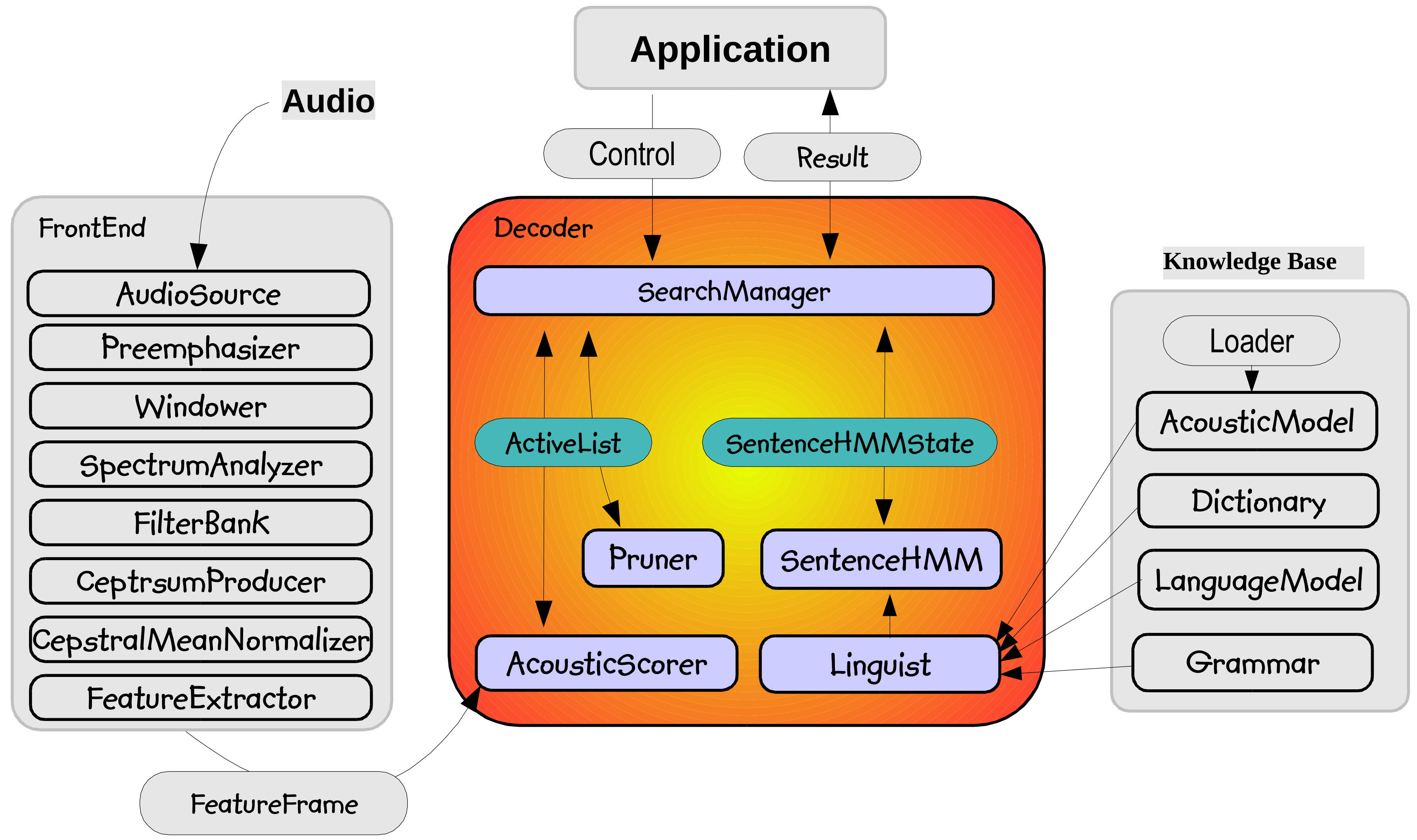
Architecture Overview



Architecture Overview

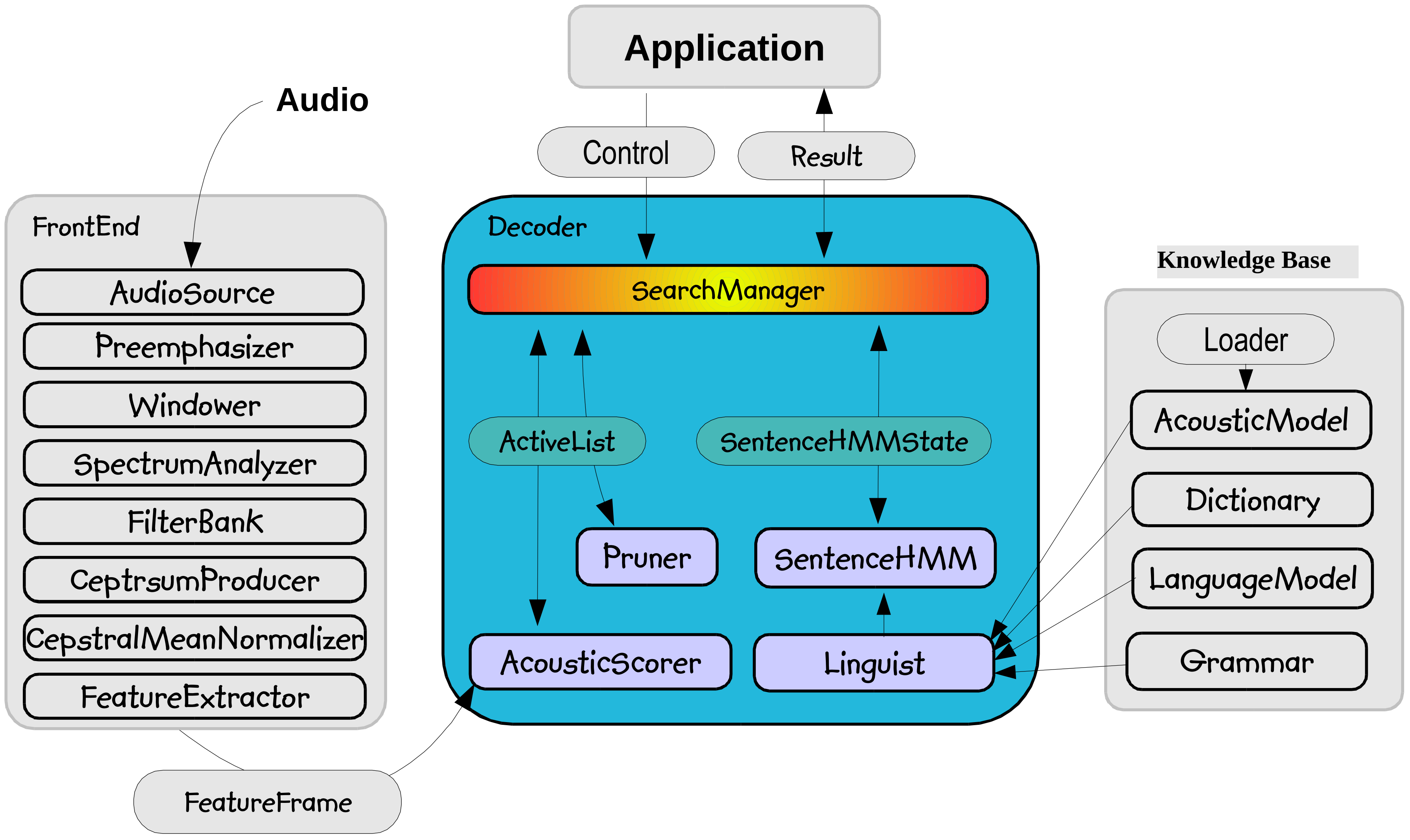


The Decoder



Application
Audio
Control
Result
FrontEnd
AudioSource
Preemphasizer
Windower
SpectrumAnalyzer
FilterBank
CeptrsumProducer
CepstralMeanNormalizer
FeatureExtractor
FeatureFrame
Decoder
SearchManager
ActiveList
SentenceHMMState
Pruner
SentenceHMM
AcousticScorer
Linguist
Knowledge Base
Loader
AcousticModel
Dictionary
LanguageModel
Grammar

SearchManager



SearchManager

- Drives the recognition process
- Relies on the SentenceHMM and the AcousticScorer
- Generates Results
- Primary Implementation is the BreadthFirstSearchManager

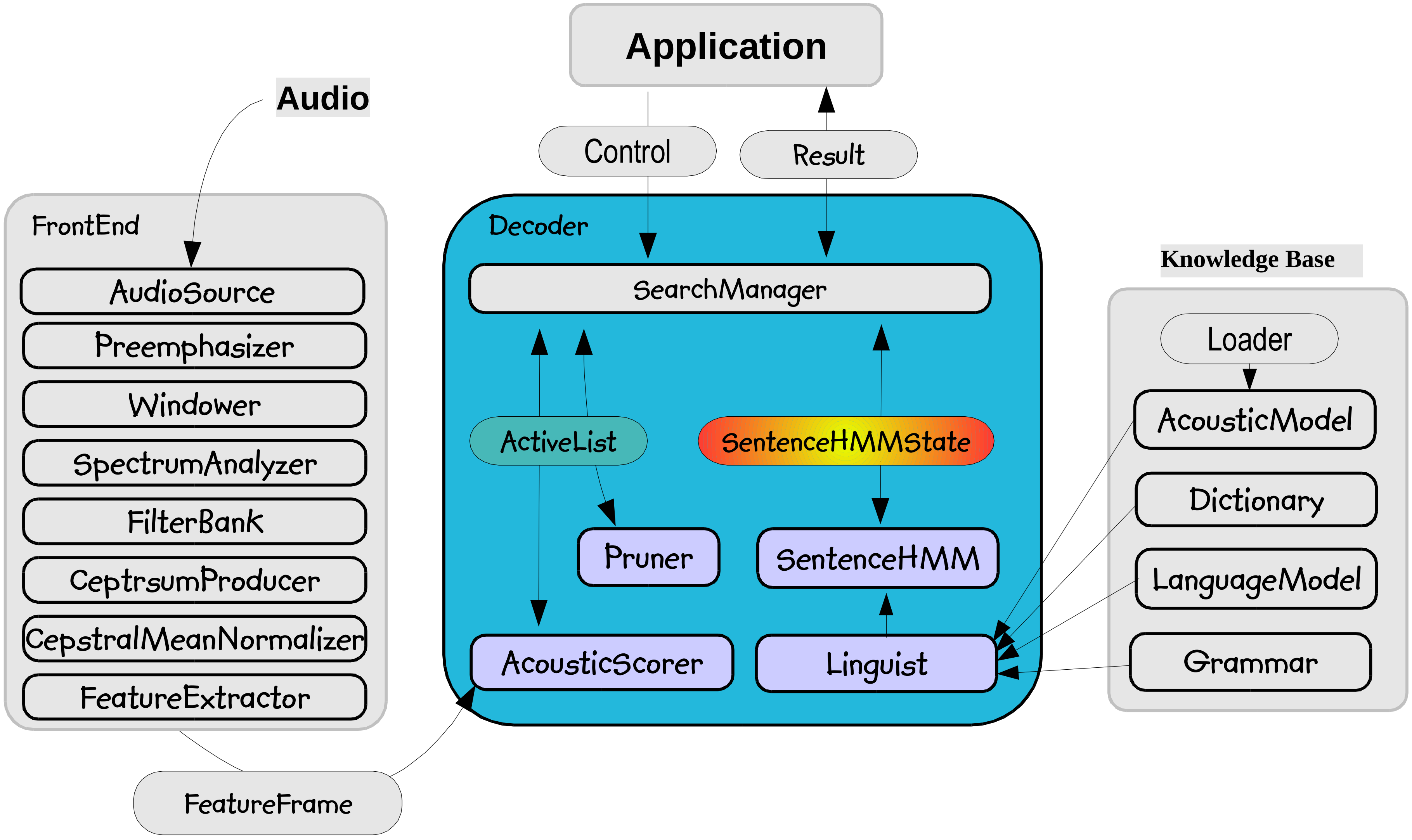
BreadthFirstSearchManager

- For each frame:
  - Scores Tokens in the ActiveList
  - Prunes Tokens from ActiveList
  - Generates Results
  - Generates next ActiveList from the SentenceHMM
- Lets look at the code

SearchManager Objects

- Search Manager uses:
  - SentenceHMMState / SentenceHMMStateArc
  - Tokens
  - The ActiveList
- Search Manager generates
  - Results

SentenceHMMState



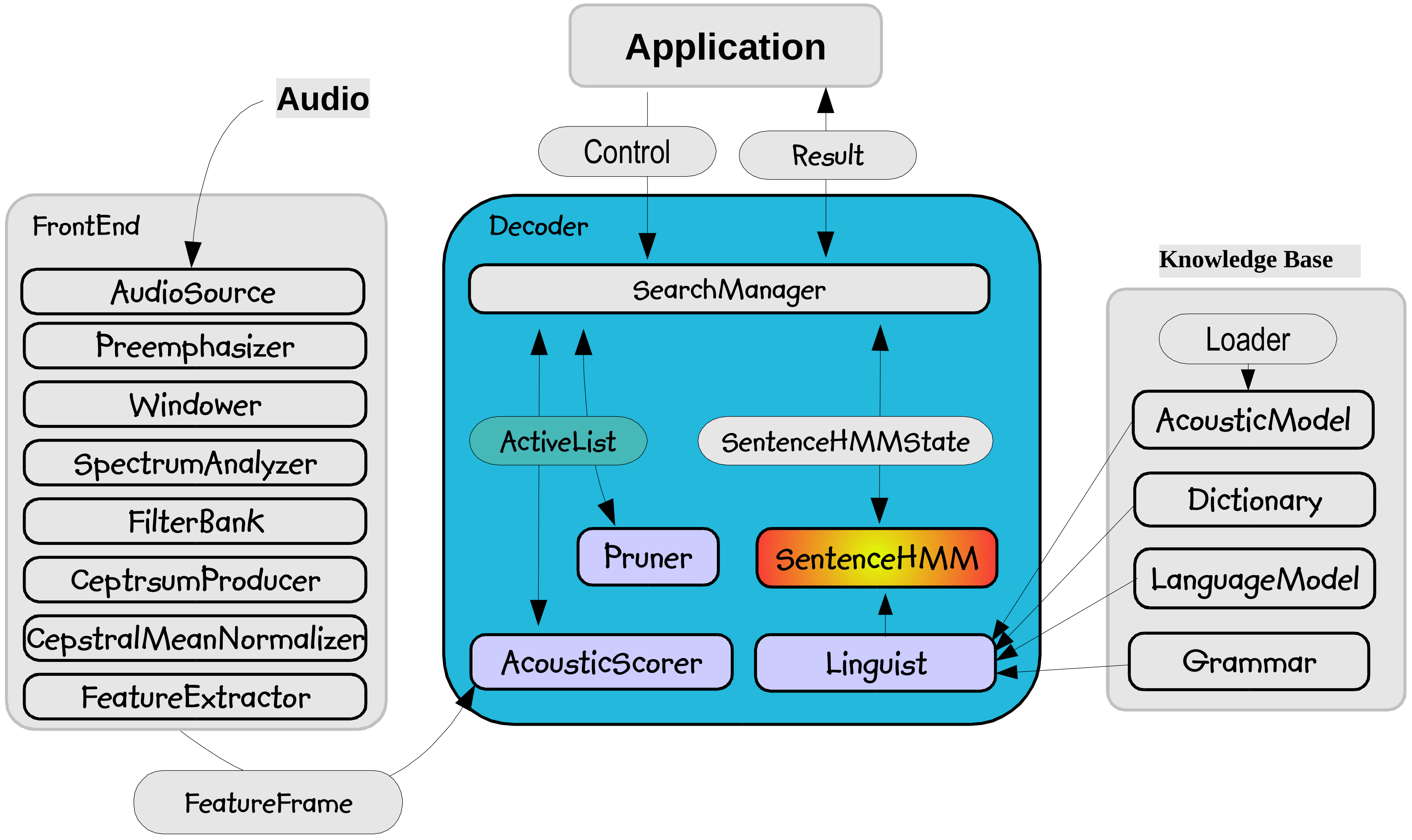
SentenceHMMState

- Represents a single state in the SentenceHMM graph
- Contains
  - Set of arcs to next SentenceHMMState
  - House keeping information
- Lets look at the code

SentenceHMMState subclasses

- SentenceHMMState is extended:
  - GrammarState
  - AlternativeState
  - WordState
  - PronunciationState
  - UnitState
  - HMMStateState

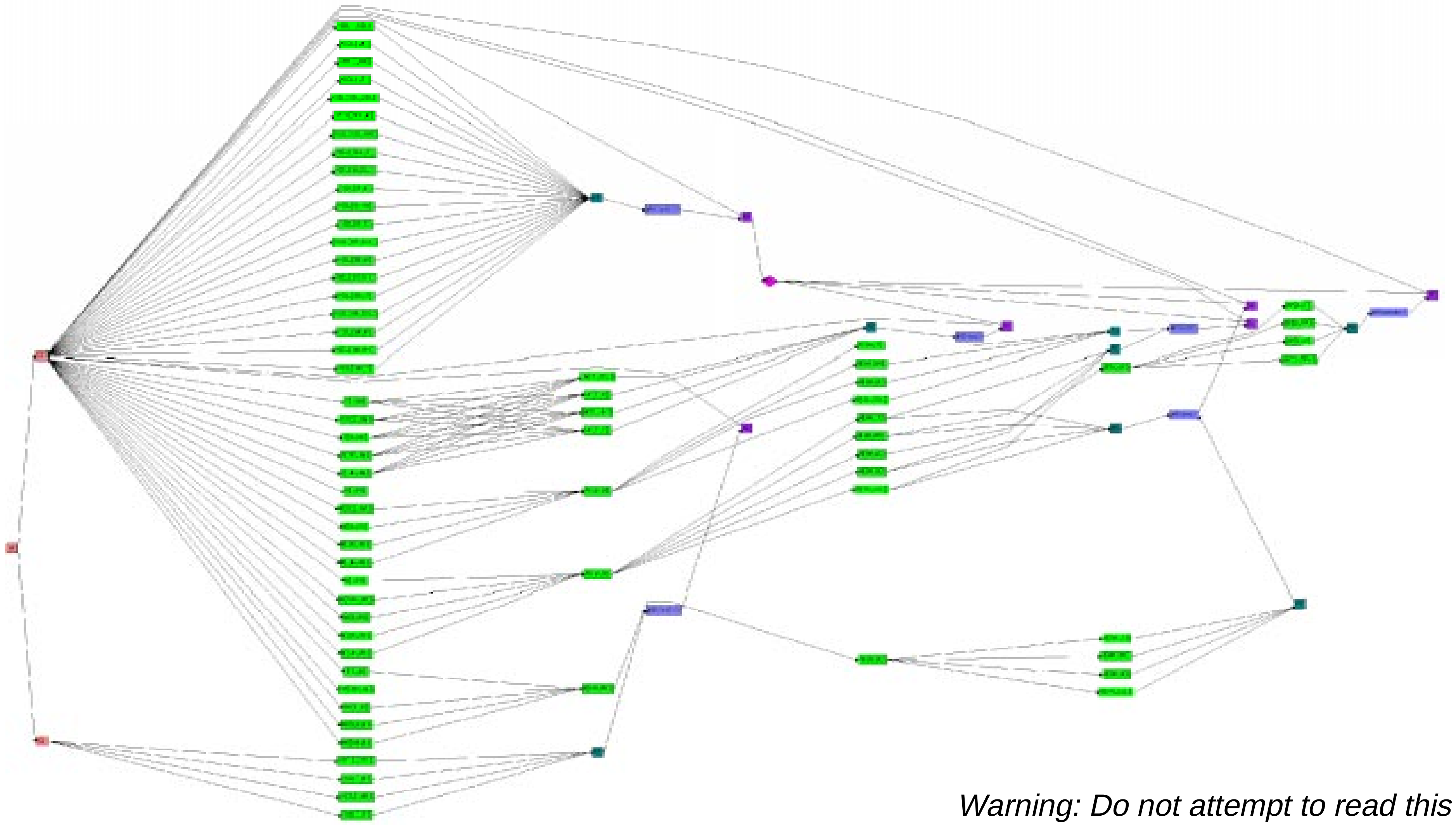
SentenceHMM



SentenceHMM

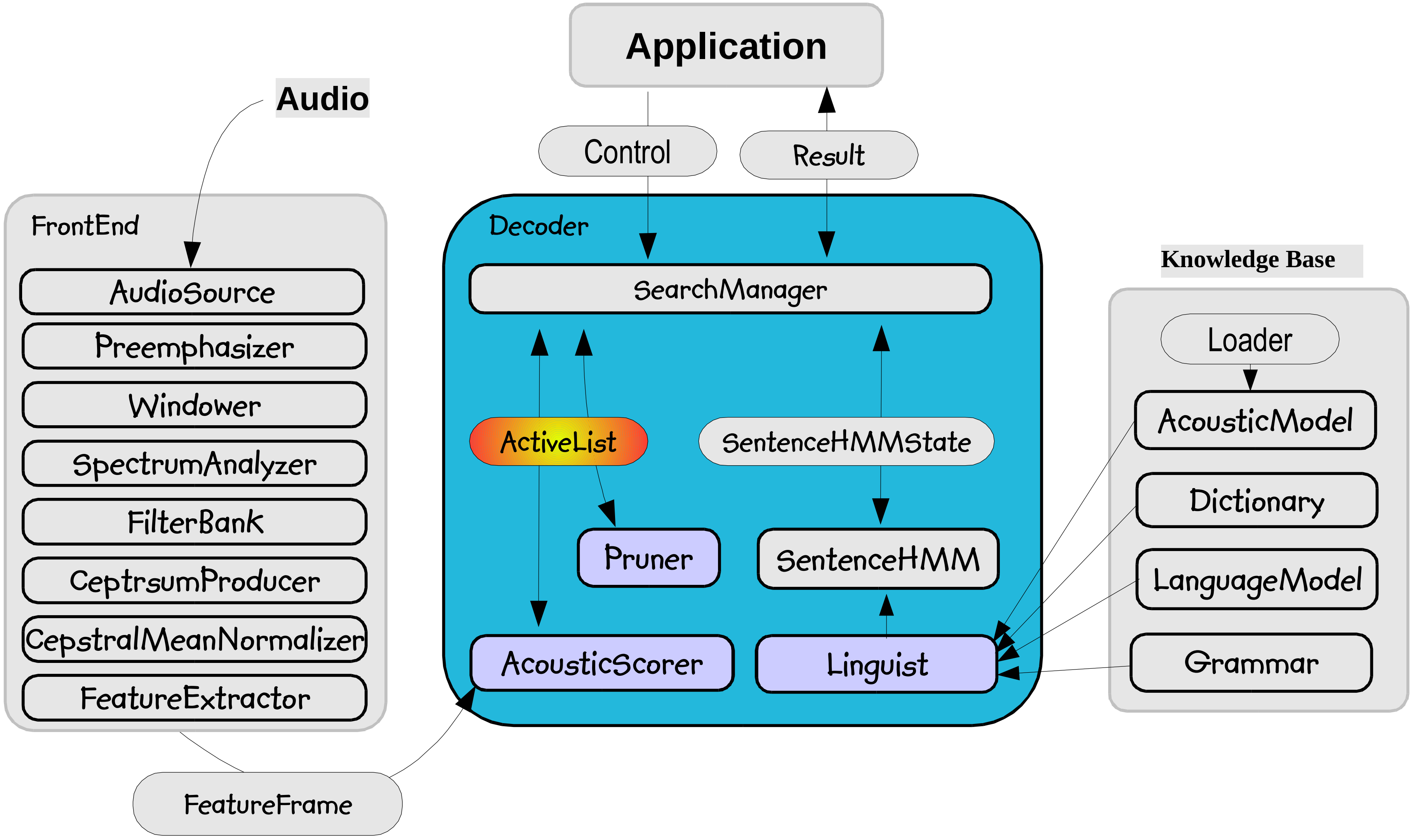
- Consists of:
  - SentenceHMMStates (and subclasses)
  - Arcs connecting these states
  - Probabilities (language, acoustic and insertion) associated with the arcs
- Defined by a single initial SentenceHMMState

Sample SentenceHMM



Warning: Do not attempt to read this

ActiveList


Application
Audio
Control
Result
FrontEnd
AudioSource
Preemphasizer
Windower
SpectrumAnalyzer
FilterBank
CeptrsumProducer
CepstralMeanNormalizer
FeatureExtractor
FeatureFrame
Decoder
SearchManager
ActiveList
SentenceHMMState
Pruner
SentenceHMM
AcousticScorer
Linguist
Knowledge Base
Loader
AcousticModel
Dictionary
LanguageModel
Grammar

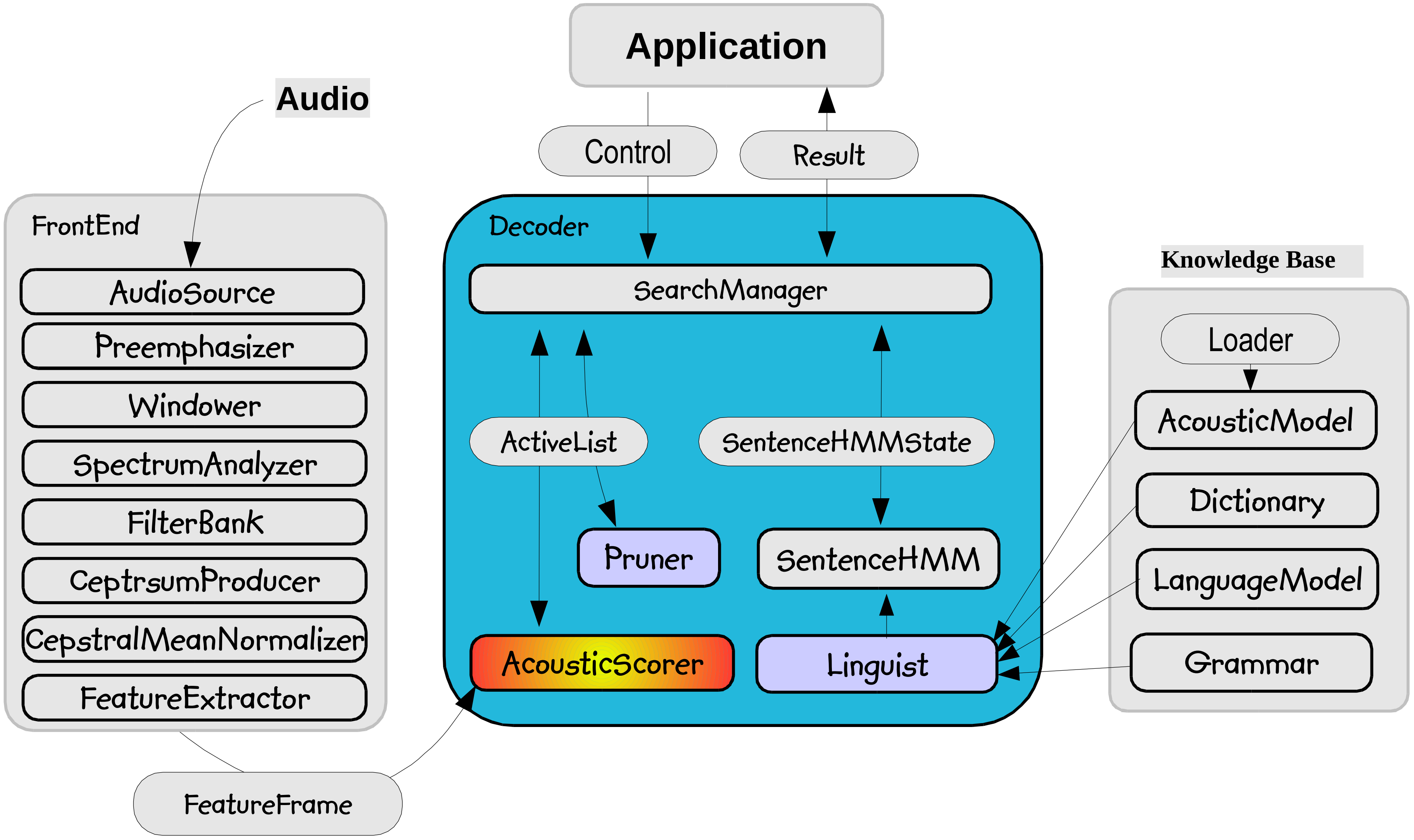
ActiveList

- Maintains list of current active tokens
- Simple Interface:
  add, replace, purge, iterator
- Implementations:
  - SimpleActiveList
  - FastActiveList
- Lets look at the code

Token

- Represents a single recognition state in the trellis for a particular frame
- Found in ActiveLists and Results
- Contains:
  - Frame number
  - Reference to SentenceHMMstate
  - Reference to previous token
  - Scoring information
- Lets look at the code

AcousticScorer



AcousticScorer

- Interface for scoring tokens
- Scores ActiveList of tokens
- Several implementations:
  - SimpleAcousticScorer
  - ThreadedAcousticScorer
- Lets look at the code

SimpleAcousticScorer

- Gets the next feature from the Front End
- Iterates through the tokens in the active list and scores the associated HMM state against the feature
- Lets Look at the code

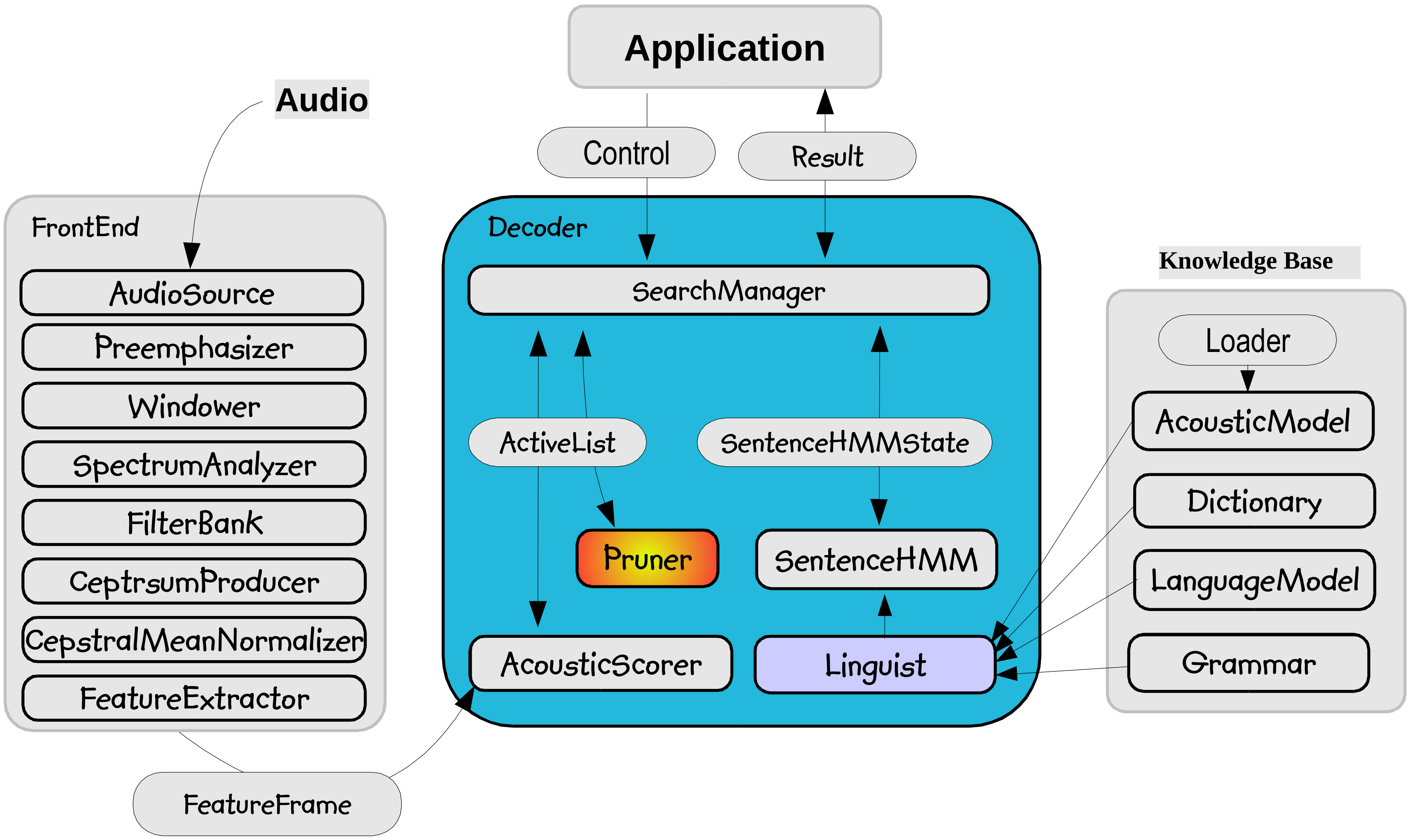
ThreadedAcousticScorer

- Creates set of scoring threads that wait on a single queue
- Breaks the active list down into small chunks and posts the chunks to the queue
- Waits for threads to score tokens

Scoring tokens

- Actual scoring code lives in
  - GaussianMixture
  - MixtureComponent
  - Uses tricksy math that Evandro will explain

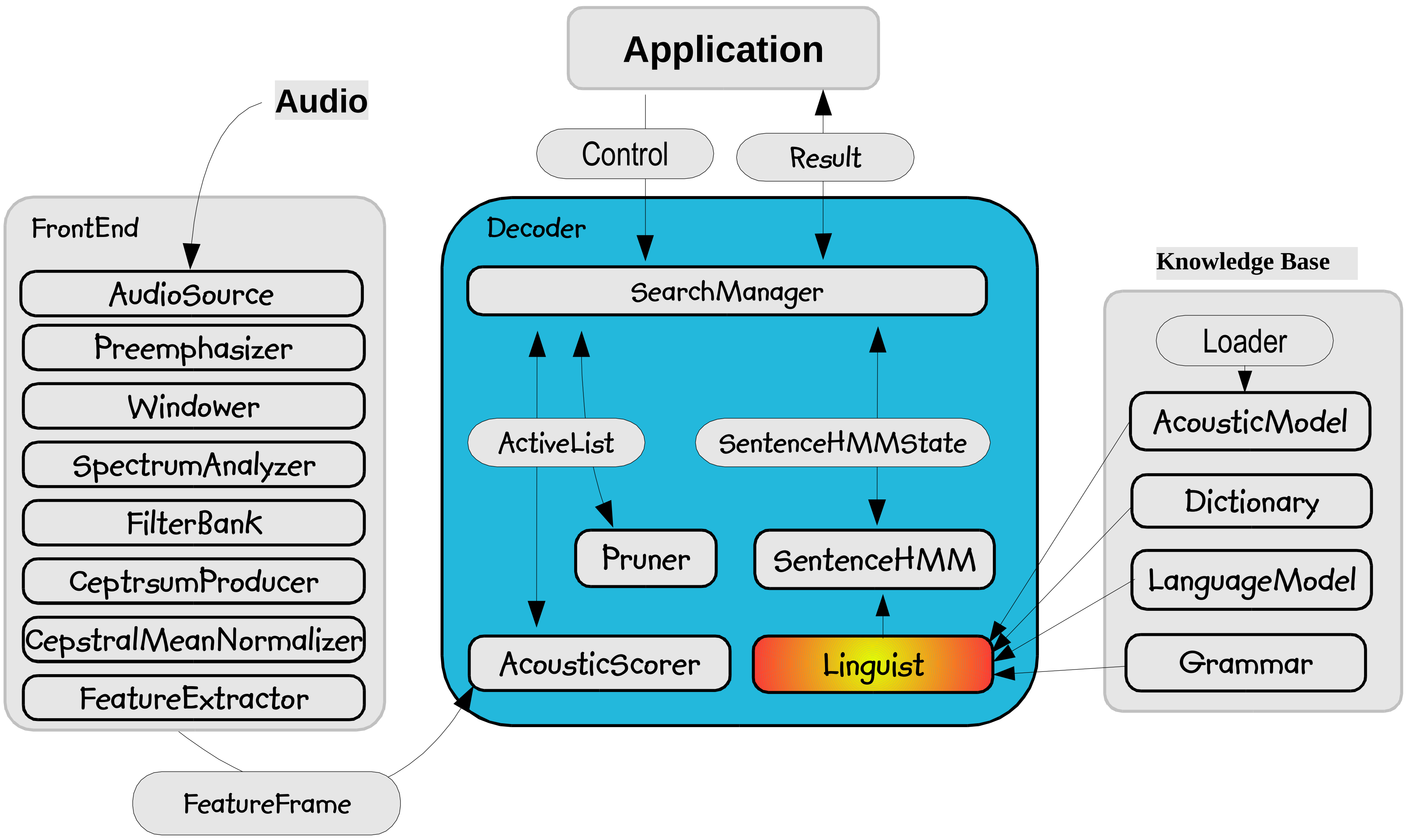
Pruner


Application
Control
Result
Audio
FrontEnd
AudioSource
Preemphasizer
Windower
SpectrumAnalyzer
FilterBank
CeptrsumProducer
CepstralMeanNormalizer
FeatureExtractor
FeatureFrame
Decoder
SearchManager
ActiveList
SentenceHMMState
Pruner
SentenceHMM
AcousticScorer
Linguist
Knowledge Base
Loader
AcousticModel
Dictionary
LanguageModel
Grammar

The Pruner

- Simple Interface for pruning the active list
- Provides mechanism for modifying pruning behavior
- Current implementations:
  - NullPruner – does nothing
  - SimplePruner – Delegate to the ActiveList.purge
- Lets look at the code

The Linguist



The Linguist

- Interface for creating the SentenceHMM
- Uses the Grammar, Acoustic Model, Dictionary and Language Model from the Knowledge base
- Several implementations, current best is   the StaticLinguist

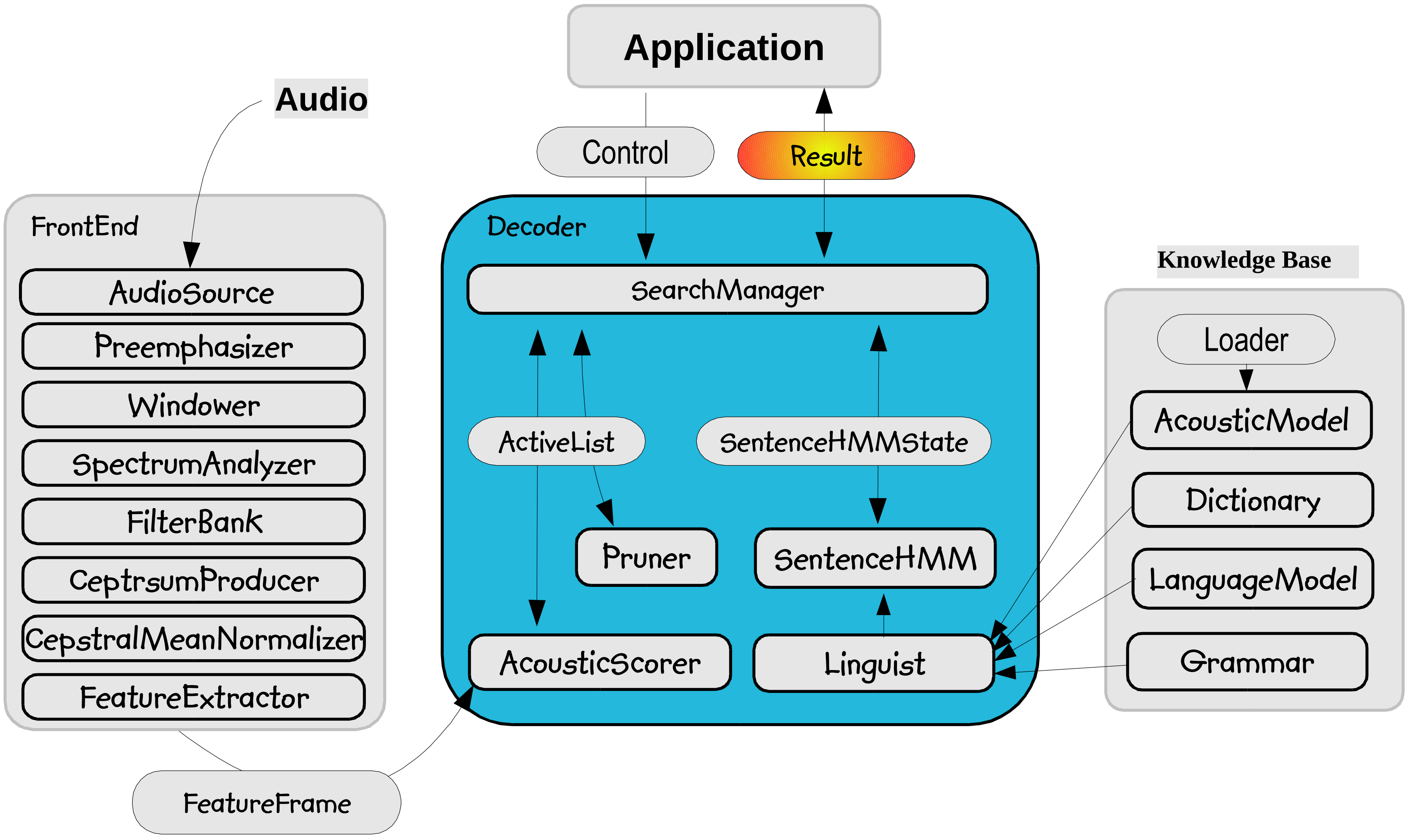
Linguist

- Uses Grammar to identify word transitions.
- Uses Dictionary to get word pronunciations
- Uses AcousticModel to get HMMs

StaticLinguist

- Generates the SentenceHMM at initialization time
- Deals with arbitrary-sized contexts
- Provides options such as:
  - Controlling fan-in
  - Flat vs. Tree layout
- A Fairly complex bit of code
- Lets look at the code

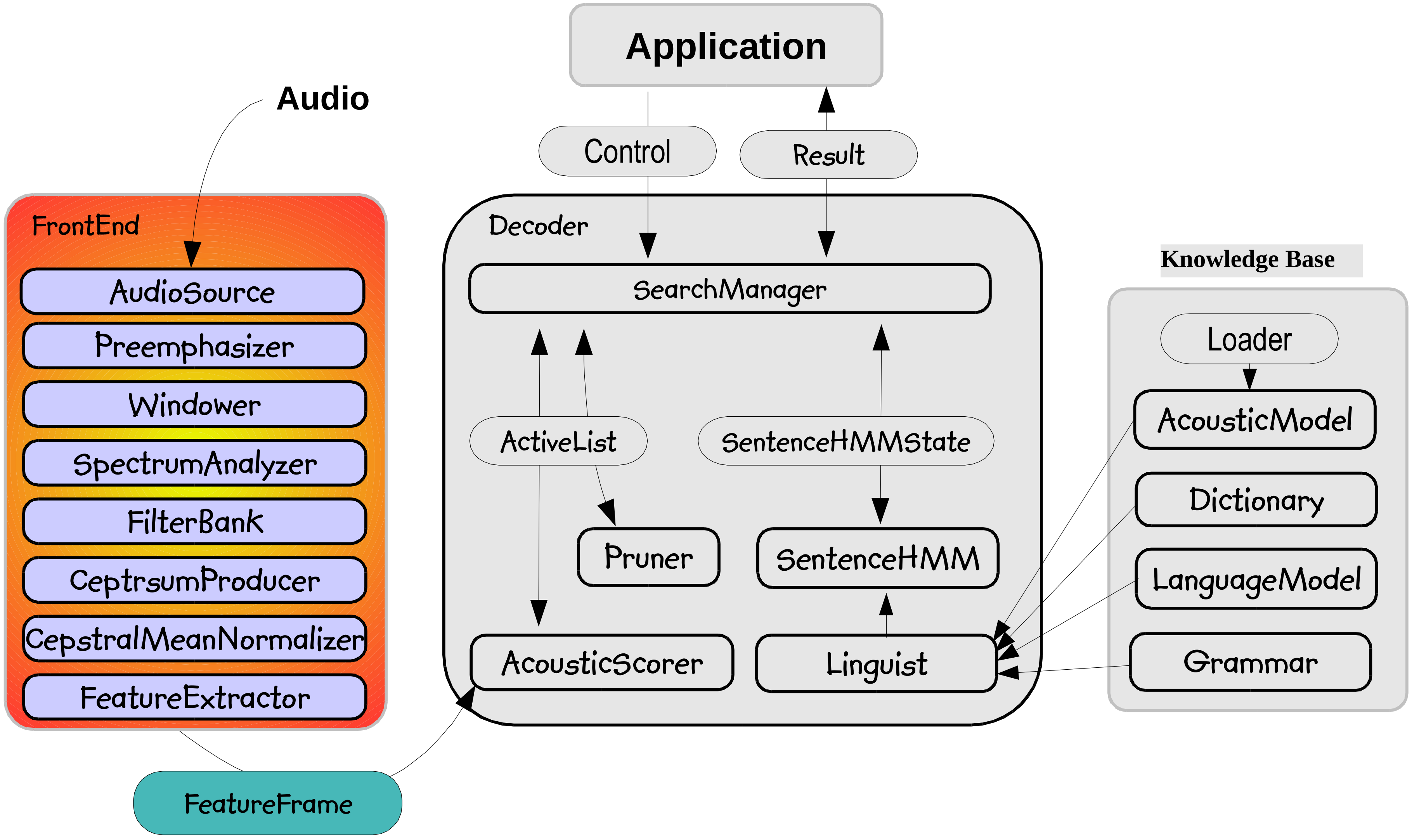
The Result


Application
Audio
Control
Result
FrontEnd
AudioSource
Preemphasizer
Windower
SpectrumAnalyzer
FilterBank
CeptrsumProducer
CepstralMeanNormalizer
FeatureExtractor
FeatureFrame
Decoder
SearchManager
ActiveList
SentenceHMMState
Pruner
SentenceHMM
AcousticScorer
Linguist
Knowledge Base
Loader
AcousticModel
Dictionary
LanguageModel
Grammar

Result

- Contains:
  - List of final state tokens
  - List of currently active tokens
  - isFinal flag
- From Result apps can derive
  - Hypothesis, N-Best list, Word timing info
- Lets look at the code

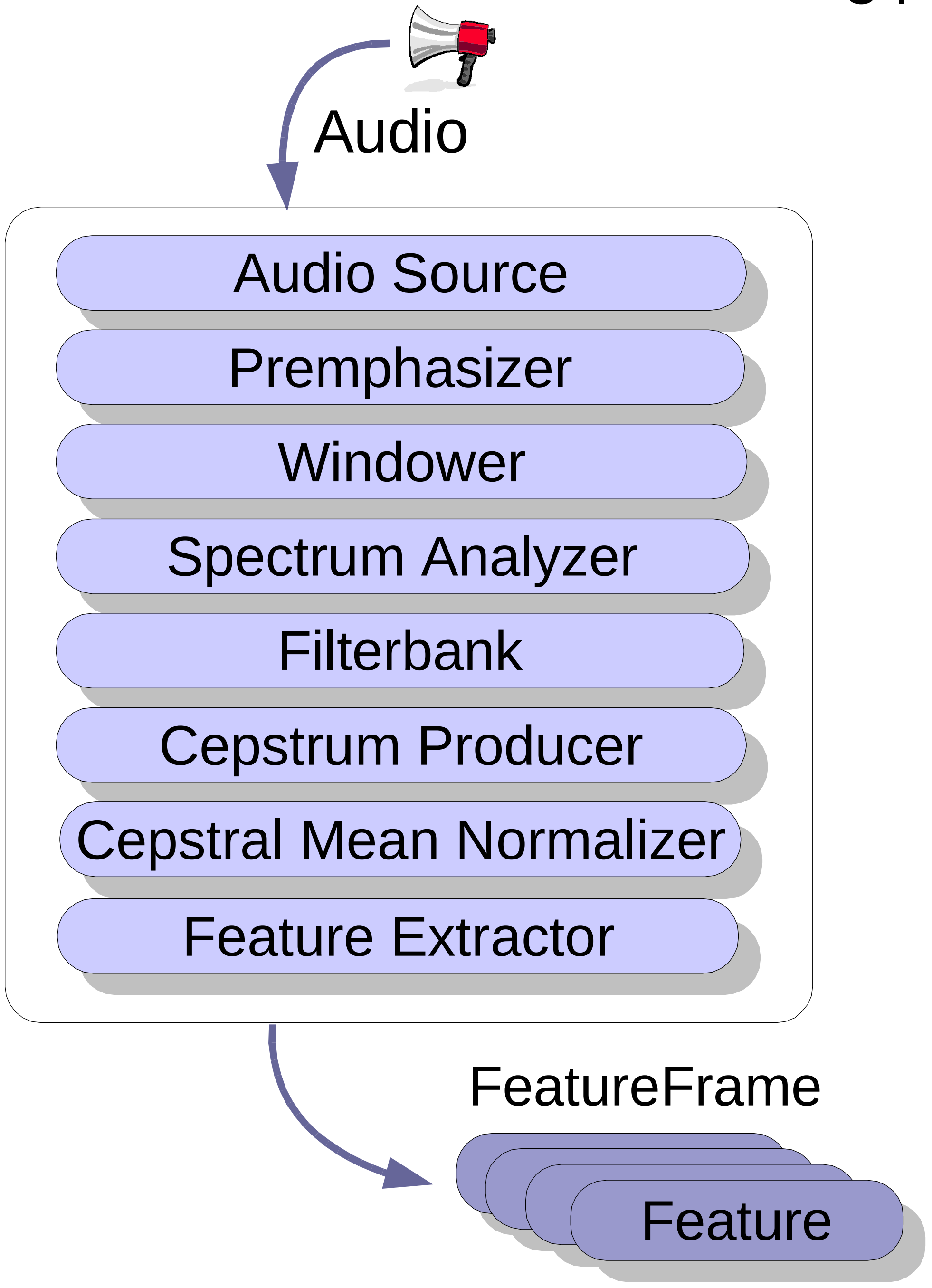
The Front End



Front-End

- speech --> features
- Front-End is a set of signal processing filters
- Simple interface:

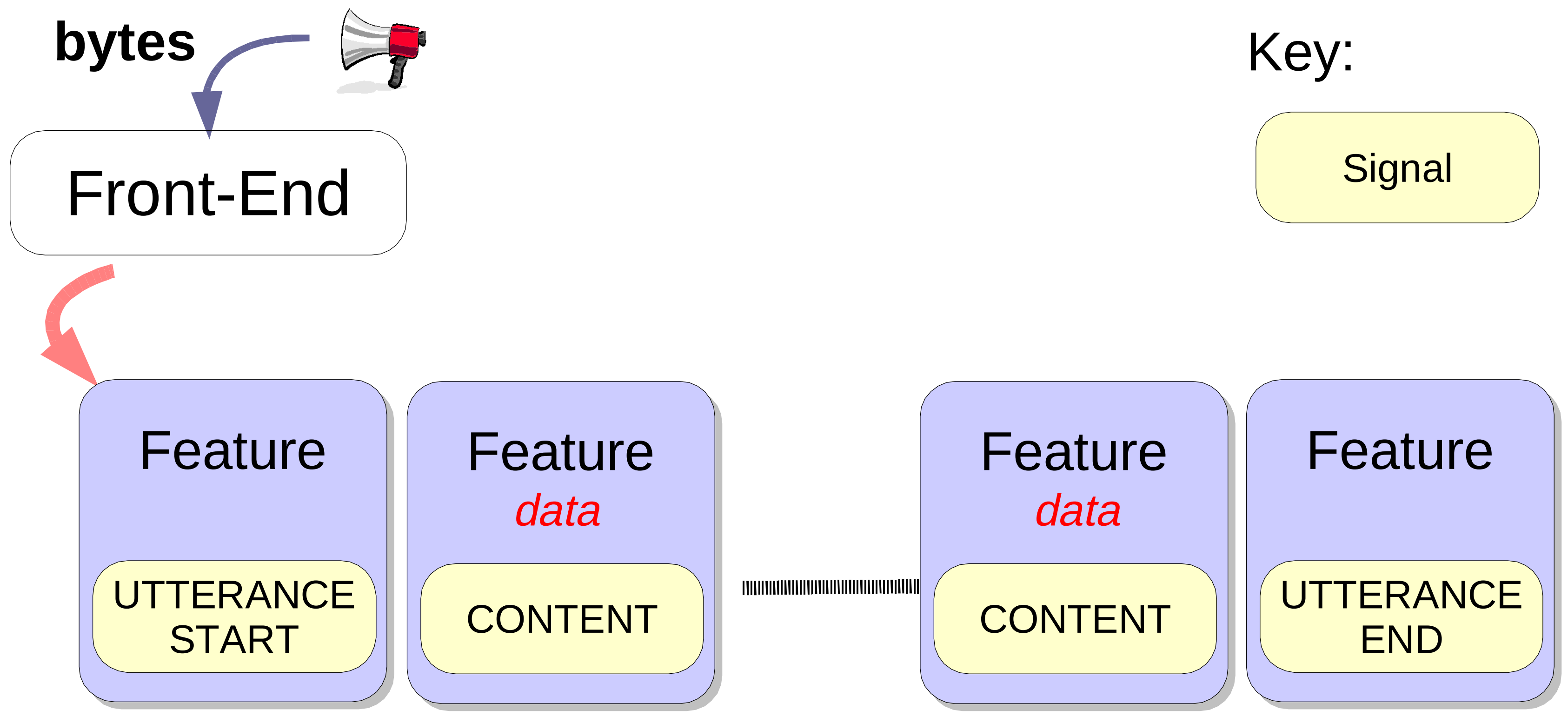
`getFeatureFrame(N)`



Data Objects

- Data objects
  - Subclasses:
    - Audio
    - Spectrum
    - Cepstrum
    - Feature
  - Contains a Signal, examples:
    - UTTERANCE_START, UTTERANCE_END
    - CONTENT (e.g., audio data)

Front-End Output

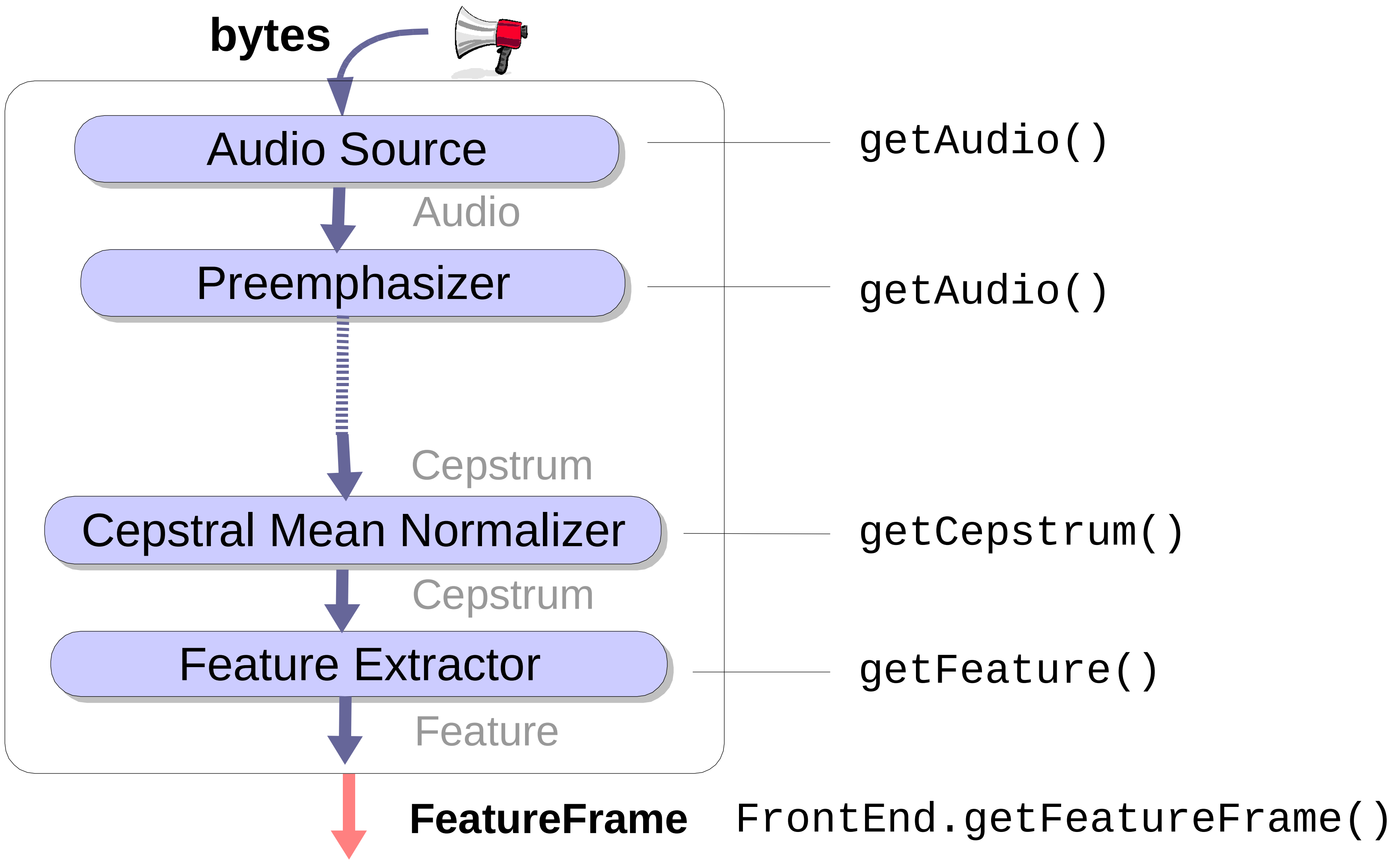


- Features of an utterance are enclosed by UTTERANCE_START and UTTERANCE_END signals.

Major Interfaces

- All front-end processors implement one of:
  - AudioSource : getAudio()
    - e.g., Preemphasizer, Windower
  - SpectrumSource : getSpectrum()
    - e.g., SpectrumAnalyzer, Filterbank
  - CepstrumSource : getCepstrum()
    - e.g. CepstrumProducer, BatchCMN
  - FeatureSource : getFeature()
    - e.g. FeatureExtractor

Front-End Pull Mechanism

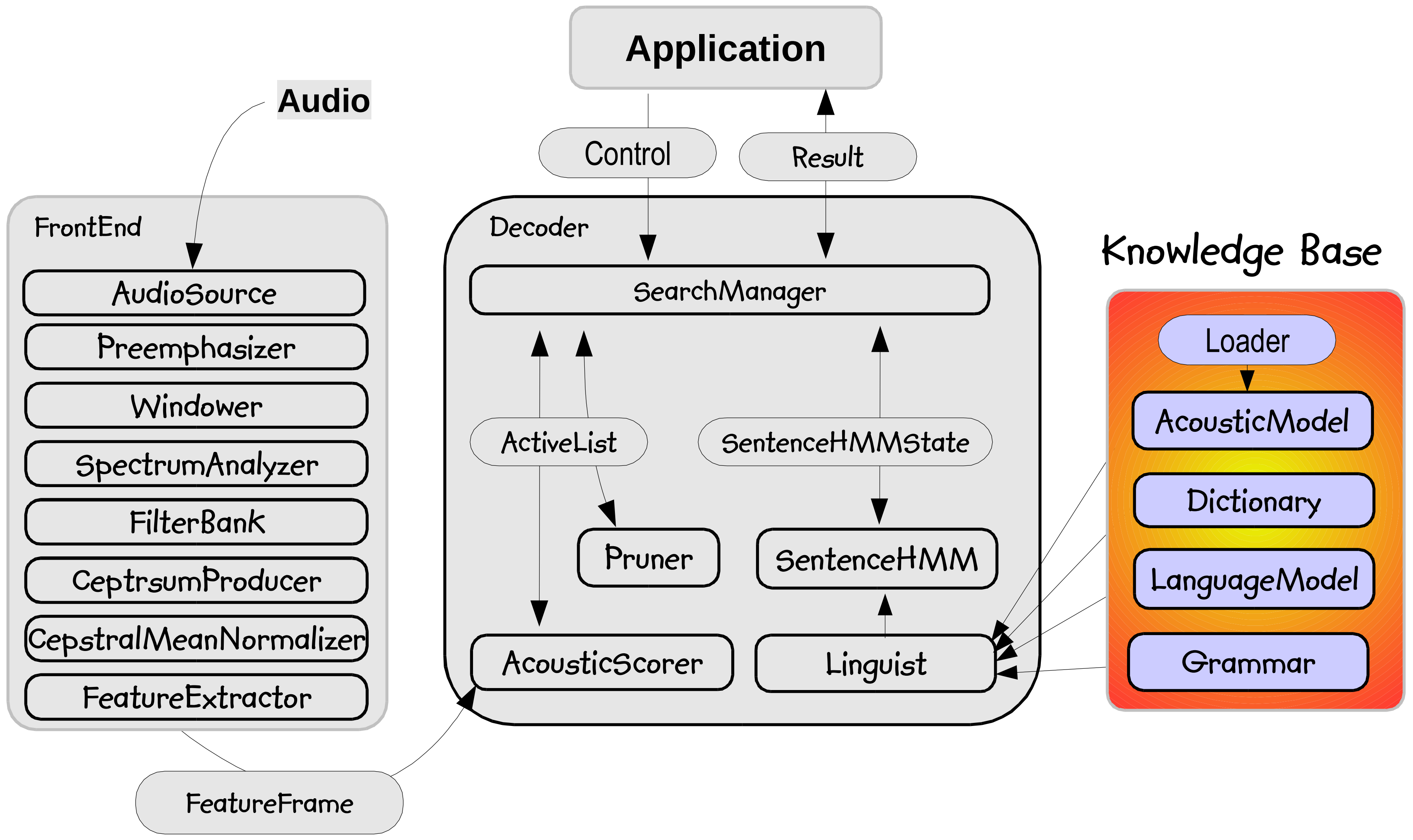


- Calling `FrontEnd.getFeatureFrame()` starts pulling.

SimpleFrontEnd

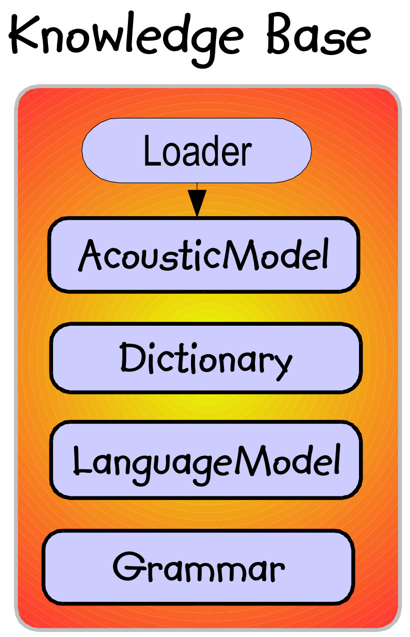
- Implements FrontEnd interface.
- Puts all the front-end processors together.
- Can accept audio or cepstra as input.
- Look at the constructor code to see how they are stitched together.

Knowledge Base



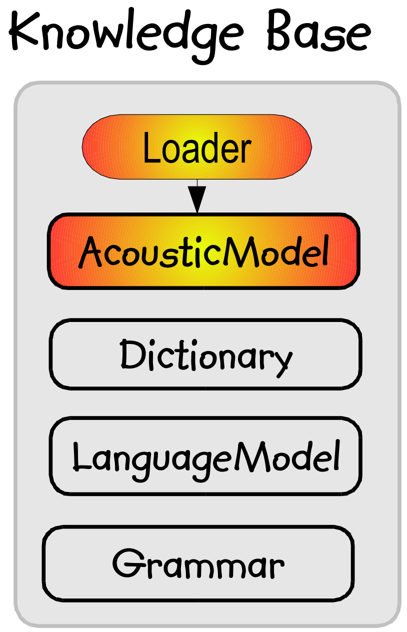
Knowledge Base

- Four disjoint sets of data
  - AcousticModel – HMMs, Gaussian Mixtures
  - Dictionary – Word pronunciations
  - Language Model – language/word transition probabilities
  - Grammar – word transitions

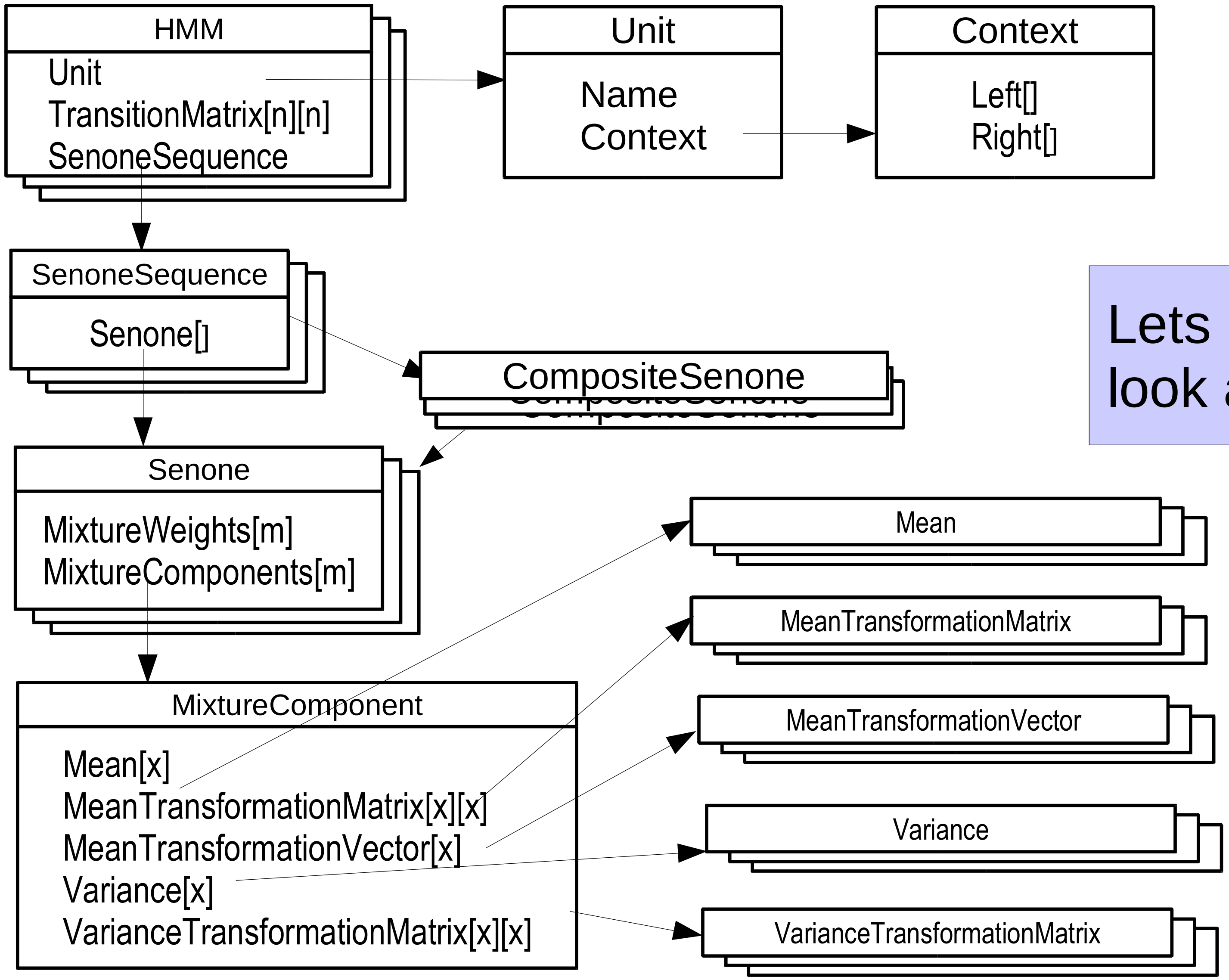


Acoustic Model

- Provides methods for looking up HMMs for a particular unit.
- A Standard 'Loader' interface provides mechanism for loading models with different formats
- Sphinx3Loader – is an implementation that loads Sphinx3 Models



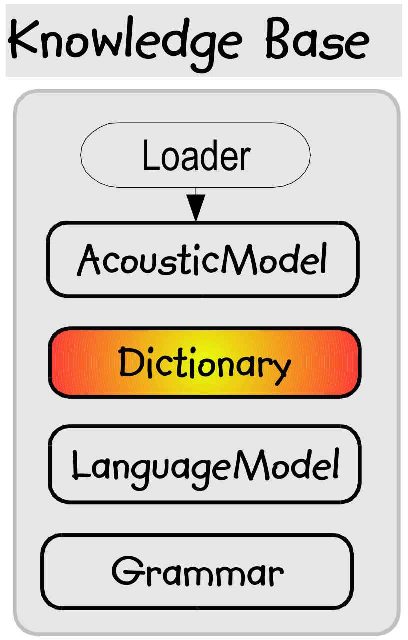
AcousticModel Layout



Lets (briefly) look at the code

Dictionary

- Standard interface
- Returns a Pronunciation for a word and WordClassification
- Currently don't do anything with WordClassification
- FullDictionary and FastDictionary are implementations



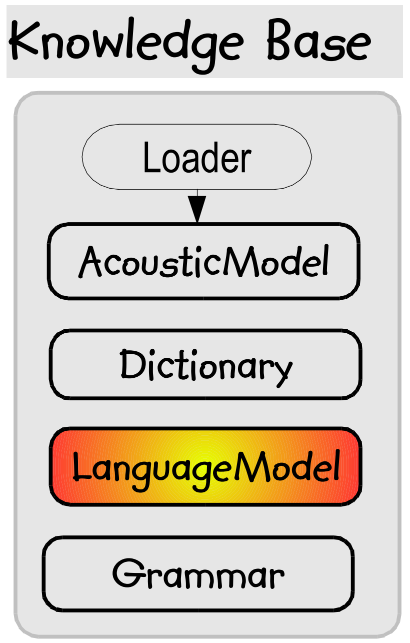
Dictionary classes

- Lets look at the code!
  - Dictionary – the interface
  - FullDictionary – original implementation – slow for small vocabulary applications
  - FastDictionary – implementation that reads CMU dictionary format
  - Pronunciation – The object returned from a lookup

The fastest I/O is no I/O.

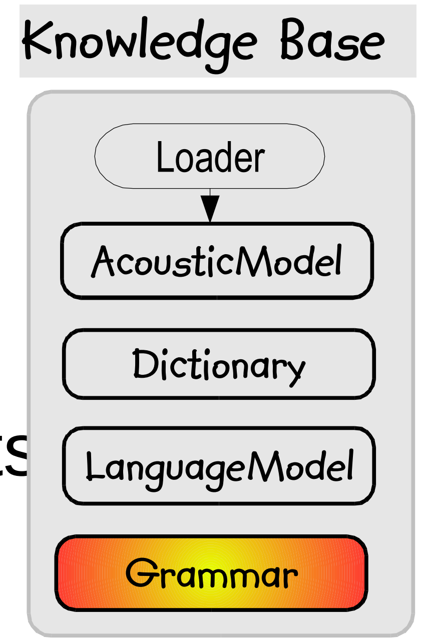
Language Model

- Provides a language probability given a word history
- Single implementation: SimpleNGramModel, loads small Sphinx3 models



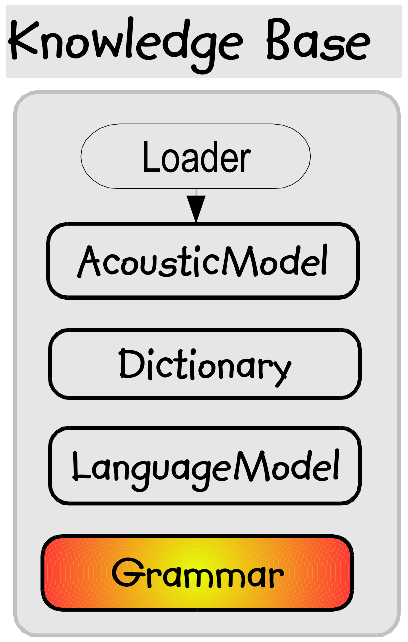
Grammar

- An abstract class that build a graph of GrammarNodes
- Several implementations:
  - WordListGrammar – simple word lists
  - ArpaGrammar – FSTS



Grammar

- Classes of interest:
  - Grammar
  - GrammarNode
  - GrammarWord
  - WordListGrammar
  - ArpaGrammar



Tools and Utilities

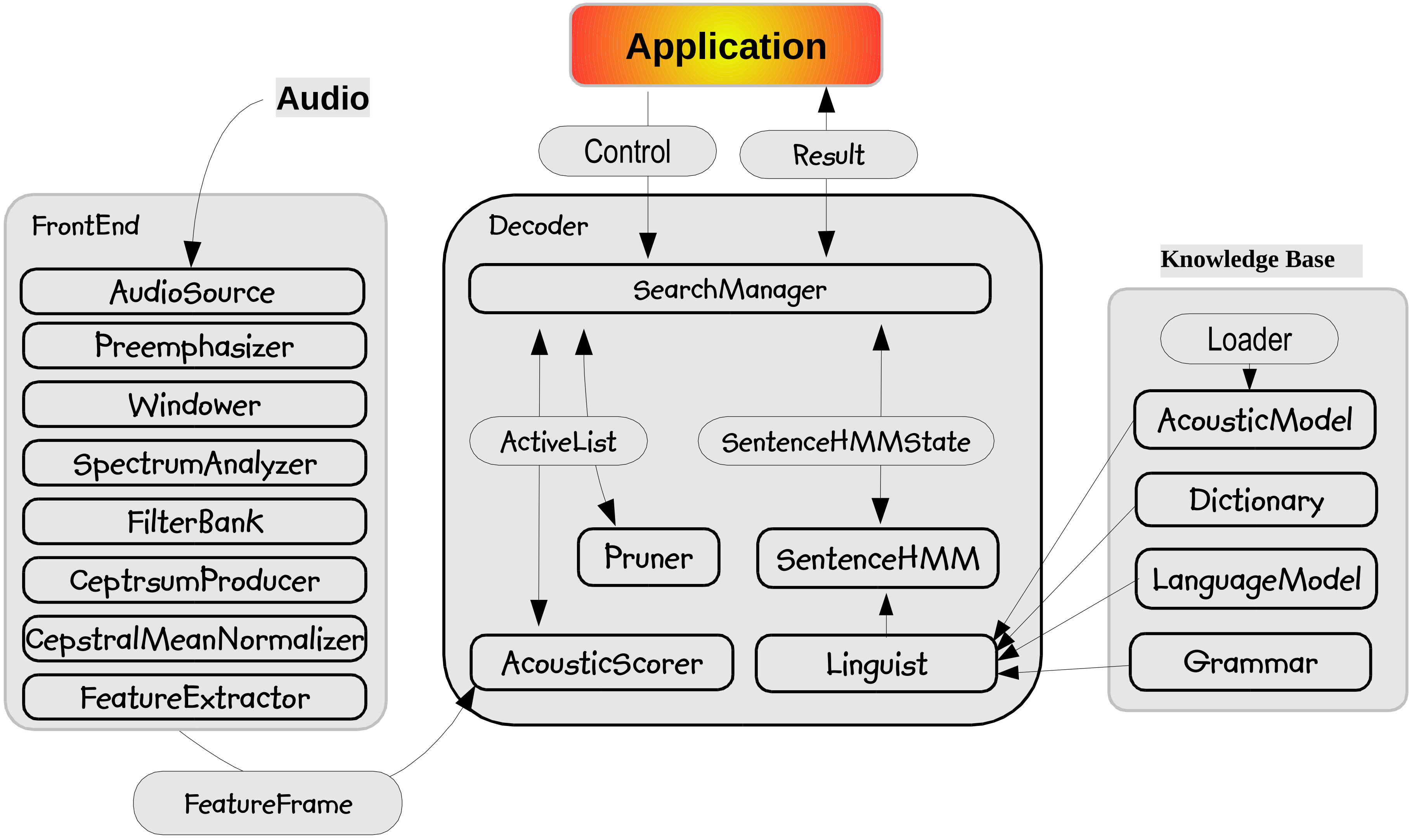
- SphinxProperties – used for configuring Sphinx 4

```
edu.cmu.sphinx.search.ActiveList.absoluteBeamWidth=800
edu.cmu.sphinx.search.ActiveList.relativeBeamWidth=1E-150
edu.cmu.sphinx.search.BreadthFirstSearchManager.filterSuccessors=false
edu.cmu.sphinx.search.BreadthFirstSearchManager.languageWeight=7.0
edu.cmu.sphinx.search.Dictionary.addSilEndingPronunciation=false
edu.cmu.sphinx.search.Linguist.expandInterNodeContexts=true
edu.cmu.sphinx.search.Linguist.showSentenceHMM=false
edu.cmu.sphinx.search.Linguist.wordInsertionProbability = 1.0E-26
edu.cmu.sphinx.search.Linguist.autoLoopSilences=false
edu.cmu.sphinx.search.StaticLinguist.isFlatSentenceHMM=false
edu.cmu.sphinx.search.validateResults=false
edu.cmu.sphinx.search.Linguist.showCompilationProgress=false
```

Tools and Utilities

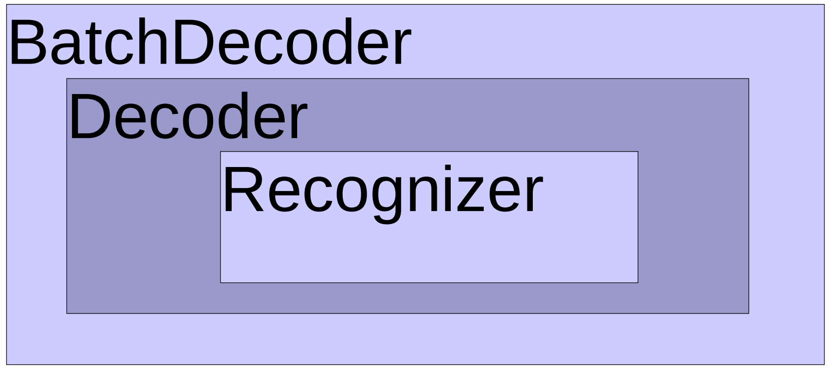
- Timer – used for timing operations
- ResultAnalyzer – calculates recognition statistics such as WER (word-error-rate)
- StatisticsVariables
- LogMath
- Logging

Application



BatchDecoder

- Recognizes audio in batch mode
- Uses Decoder to perform recognition and to show results
- Decoder uses Recognizer to select all of the components



Code Sundries

The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time.

Getting the Source Code

- Instructions for getting the code at:

  http://sourceforge.net/cvs/?group_id=1904

- Browse the source code at:
  http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/cmusphinx/

- Browse the Javadoc API at:

  http://cmusphinx.sourceforge.net/sphinx4/

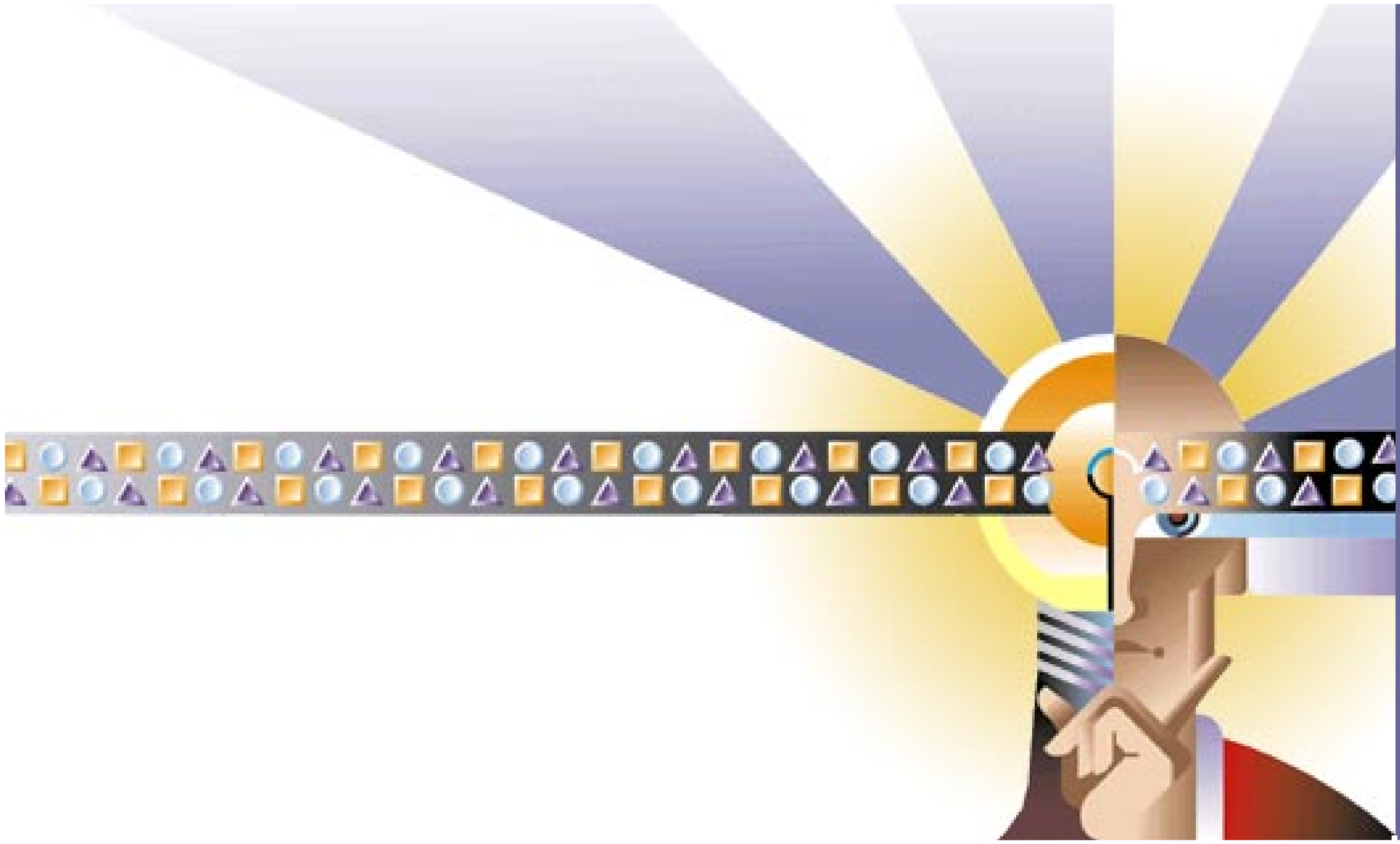
Sphinx 4 Metrics

Number of files	592
Number of source files	201
Number of classes	364
Source lines of code	16621
Packages	20

Source Tree Structure

```
|-- build
|-- doc
|-- edu
|    -- cmu
|         -- sphinx
|            |-- decoder
|            |-- frontend
|            |    |-- mfc
|            |    |-- parallel
|            |    |-- plp
|            |    `-- processors
|            |-- jsapi
|            |-- model
|            |    |-- acoustic
|            |    `-- language
|            |-- result
|            |-- search
|            --- util
```

```
|-- lib
|-- scripts
`-- tests
    |-- decoder
    |    `-- live
    |-- frontend
    |-- junit
    |-- live
    |-- other
    |-- performance
    |    |-- an4
    |    |-- aurora
    |    |-- benchmarks
    |    |-- rm1
    |    |-- ti46
    |    `-- tidigits
    |-- regression
    |    `-- dummyTests
    `-- search
```



Sphinx 4 Code Walkthrough

Q&A