Sphinx 3.4 Development Progress

Arthur Chan, Jahanzeb Sherwani
Carnegie Mellon University
Mar 4, 2004

This seminar

- Overview of Sphinxes (5 mins.)
- Report on Sphinx 3.4 development progress (40 mins.)
  - Speed-up algorithms
  - Language model facilities
- User/developer forum (20 mins.)

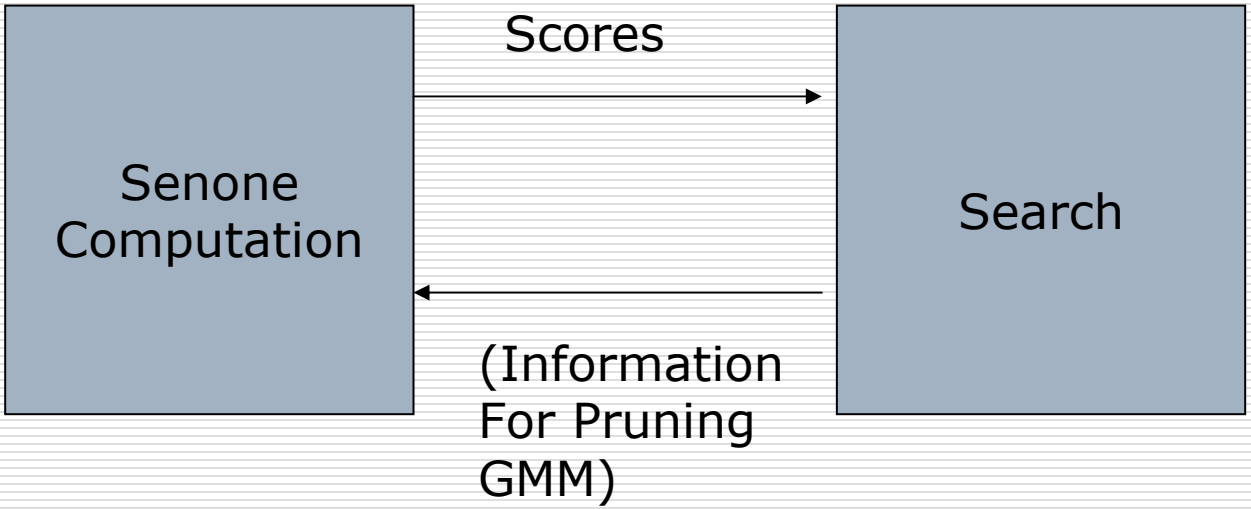
Sphinxes

- Sphinx 2
  - Semi-continuous HMM-based
  - Real-time performance : 0.5xRT – 1.5xRT
    - Tree lexicon
  - Ideal for application development
- Sphinx 3
  - Fully-Continuous HMM
  - Significantly slower than Sphinx 2 : 14-17xRT (tested in P4 1G)
    - Flat lexicon.
  - Ideal for researcher
- Sphinx 3.3
  - Significant modification of Sphinx 3
  - Close to RT performance 4-7xRT – Tree lexicon

Sphinx 3.4

- Descendant of Sphinx 3.3
  - With improved speed performance
    - Already achieved real-time performance (1.3xRT) in Communicator task.
  - Target users are application developers
  - Motivated by project CALO

Overview of S3 and S3.3: Computations at every frame



S3: -Flat lexicon, all senones are computed.
S3.3: -Tree lexicon, senones only when active in search.

Current Systems Specifications (without Gaussian Selection)

	Sphinx 3	Sphinx 3.3
Speed in P4-1G Tested in Communicator Task	ERR 17.2% 11xRT GMM, 3xRT Srch	ERR 18.6% 6xRT GMM, 1xRT Srch
GMM Computations	Not optimized (few code optimization)	Can applied Sub-VQ-based Gauss. Selection
Lexicon	Flat	Tree
Search	Beam on search, no beam on GMM	Beam on Search Beam on GMM.

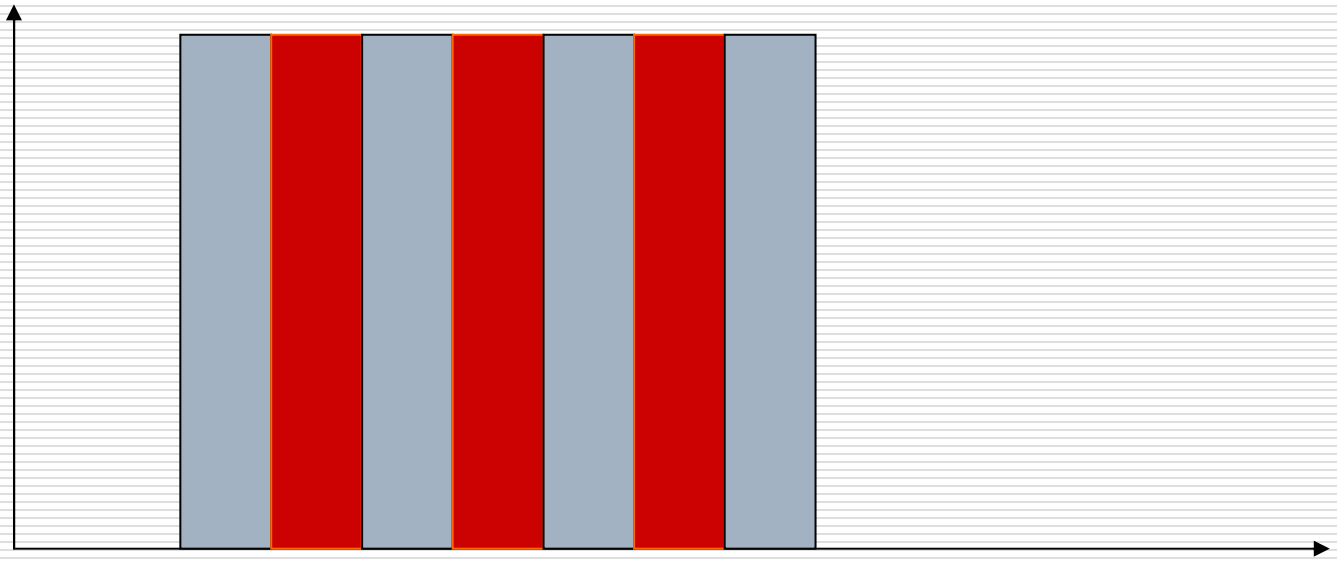
Our Plan in Q1 2004: upgrade s3.3 to s3.4

- Fast Senone Computation
  - 4-Level of Optimization
- Other improvements
  - Phoneme look-ahead
    - Reduction of search space by determining the active phoneme list at word-begin.
  - Multiple and dynamic LM facilities

Fast Senone Computation

- More than >100 techniques can be found in the literature from 1989-2003.
- Most techniques
  - claim to have 50-80% reduction of computation
  - with “negligible” degradation
    - Practically : It translate to 5% to 30% relative degradation.
- Our approaches
  - categorize them to 4 different types
  - implement representative techniques
  - tune system to <5% degradation
- Users can choose which types of technique should be used.

Fast GMM Computation: Level 1: Frame Selection



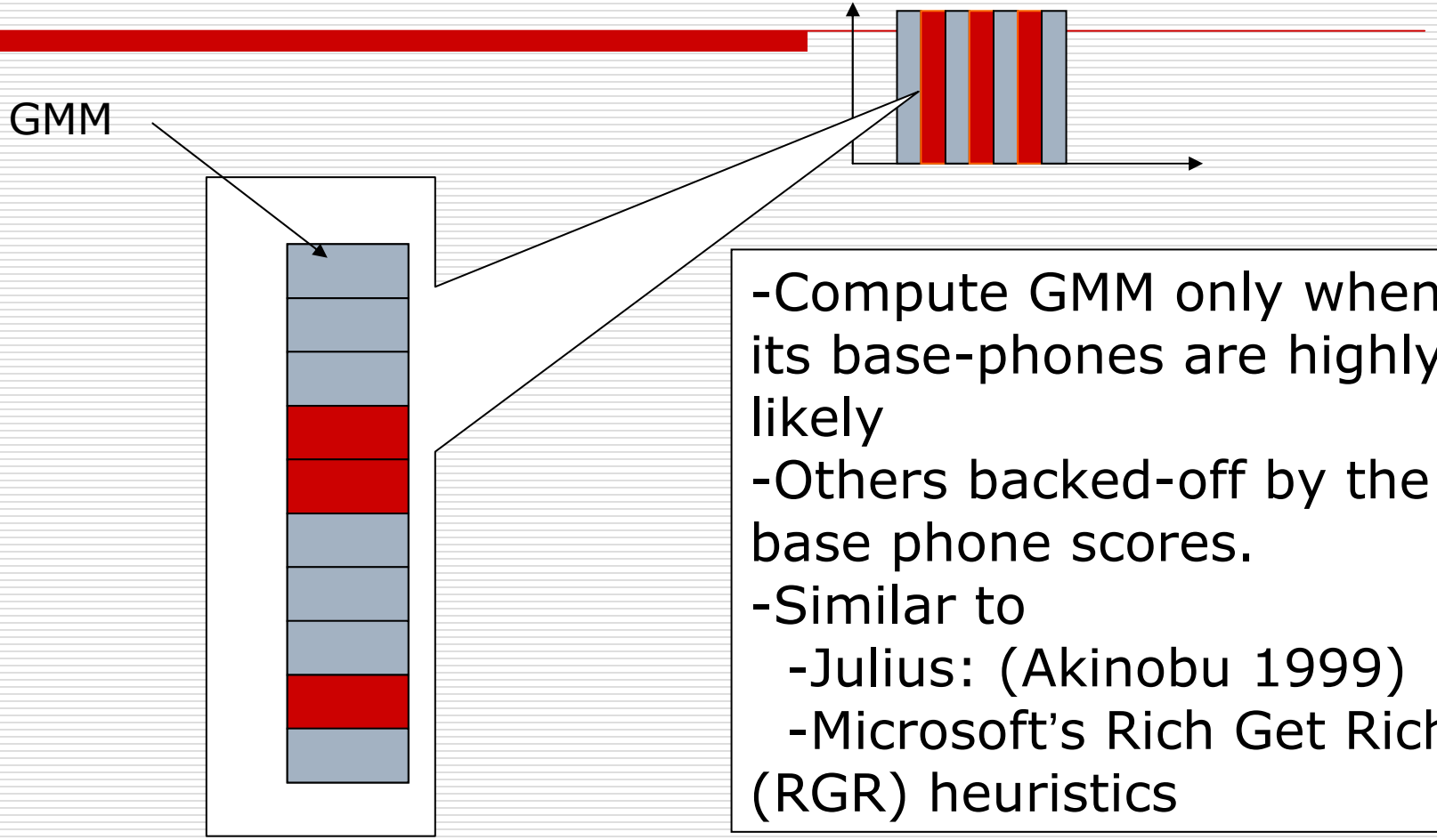
-Compute GMM in one and other frame only

-Improvement : Compute GMM only if current frame is similar to previous frame

Algorithms

- The simple way (Naïve Down-Sampling)
  - Compute senone scores only one and another N frames
- In Sphinx 3.4, implemented
  - Simple way
  - Improved version (Conditional Down-Sampling)
    - Found sets of VQ codebook.
    - If a vector is clustered to a codeword again, computation is skipped.
- Naive down-sampling
  - Rel 10% degradation, 40-50% reduction
- Conditional down-sampling
  - Rel 2-3% degradation, 20-30% reduction

Fast GMM Computation: Level 2 : Senone Selection

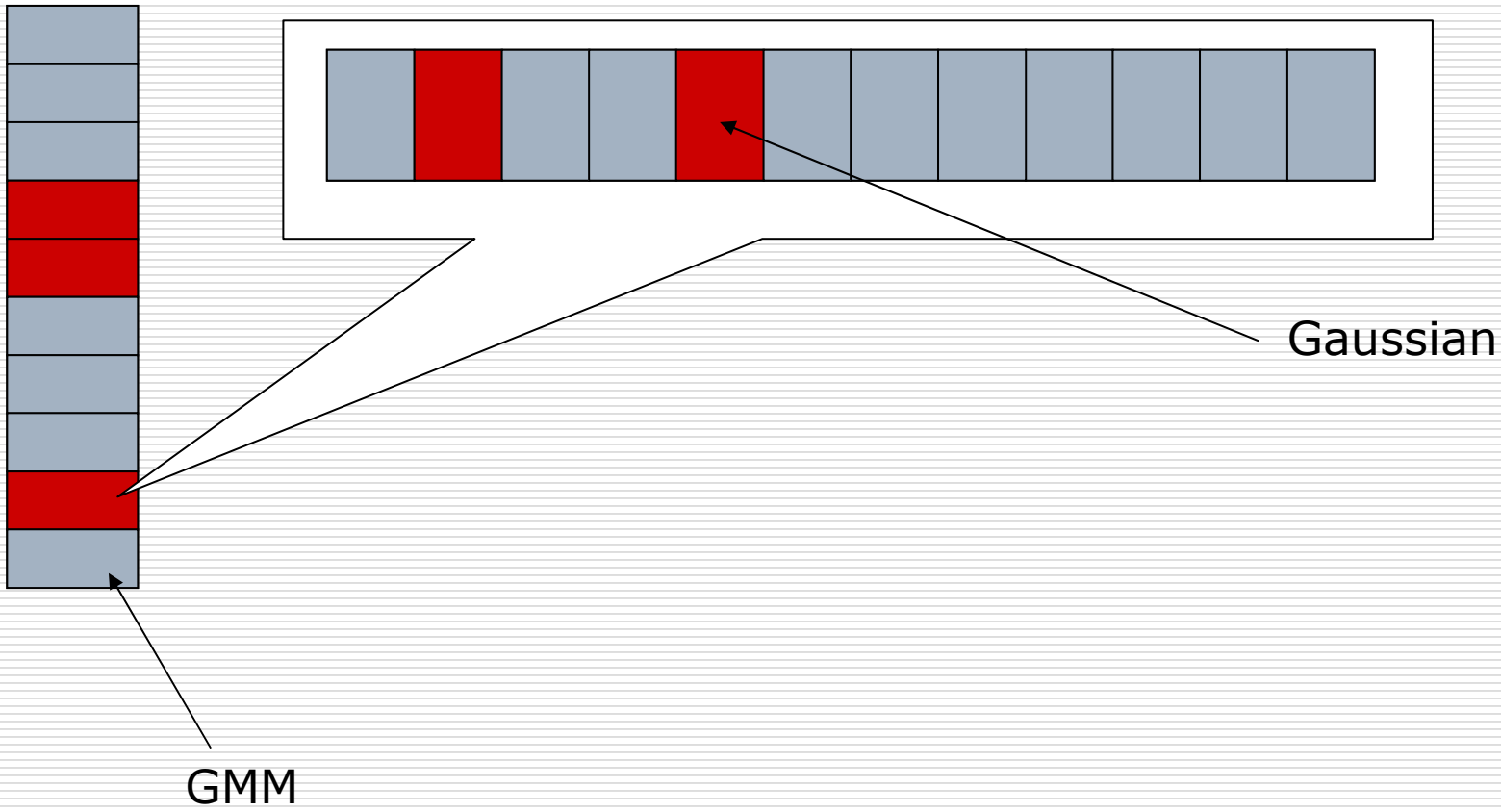


-Compute GMM only when its base-phones are highly likely
-Others backed-off by the base phone scores.
-Similar to
  -Julius: (Akinobu 1999)
  -Microsoft's Rich Get Richer (RGR) heuristics

Algorithm: CI-based Senone Selection

- If base CI senone of CD senone has high score
  - E.g. aa (base CI senone) of t_aa_b (CD senone)
  - compute CD senone
- Else,
  - Back-off to CI senone
- Known problems.
  - Back-off caused many senone scores be the same
  - Caused inefficiency of the search
- Very effective
  - 75%-80% reduction of senone computation with <5% degradation
  - Worthwhile in system with large portion time spent in doing GMM computation.

Fast GMM Computation: Level 3 : Gaussian Selection



Algorithm: VQ-based Gaussian Selection

- Bochierri 93
- In training:
  - Pre-compute a set of VQ codebook for all means.
  - Compute the neighbors for each senones for codeword.
    - If the mean of a Gaussian is closed to the codeword, consider it as a neighbor.
- In run-time:
  - Find the closest codeword for the feature.
  - compute Gaussian distribution(s) only when they is/are the neighbor
- Quite effective 40-50% reduction, <5% degrdation

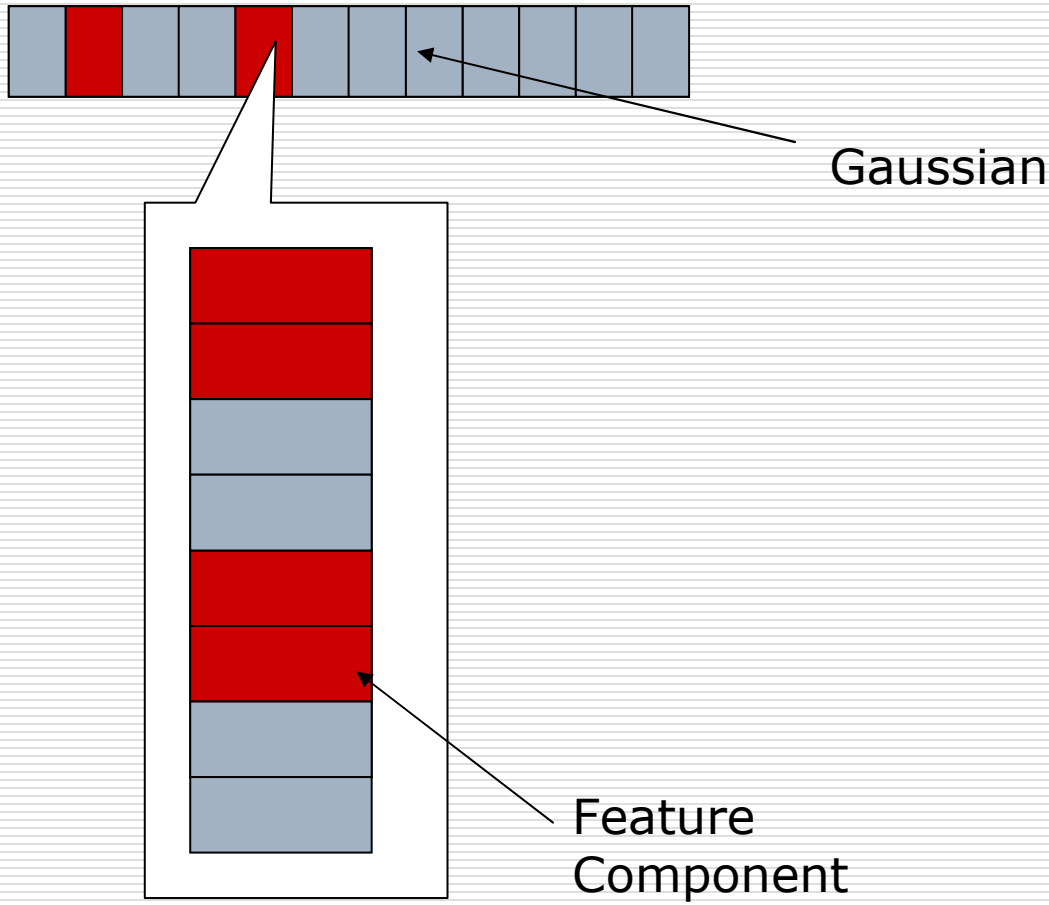
Issues:

- Require back-off schemes.
  - Minimal number of neighbors
  - Always use the closest Gaussian as a neighbor (Douglas 99)
- Further constraints to reduce computation.
  - Dual-ring constraints (Knill and Gales 97)
- Overhead is quite significant

Other approaches

- Tree-based algorithm
  - k-d tree
  - Decision tree
- Issues : How to adapt these models?
  - No problem for VQ-based technique
  - Research problems.

Fast GMM Computation: Level 4 : Sub-vector quantization



Algorithm (Ravi 98)

- In training:
  - Partition all means to subvectors
  - For each sets of subvectors
    - Find a set of VQ code-book
- In run-time:
  - For each mean
    - For each subvector
      - Compute the closest index
  - Compute Gaussian score by combining all subvector scores.

Issue

- Can be used in Gaussian Selection
  - Use approximate score to decide which Gaussian to compute
- Use as an approximate score
  - Require large number of sub-vectors (13)
  - Overhead is huge
- Use as Gaussian Selection
  - Require small amount of sub-vectors(3)
  - Overhead is still larger than VQ.
    - Machine-related issues.

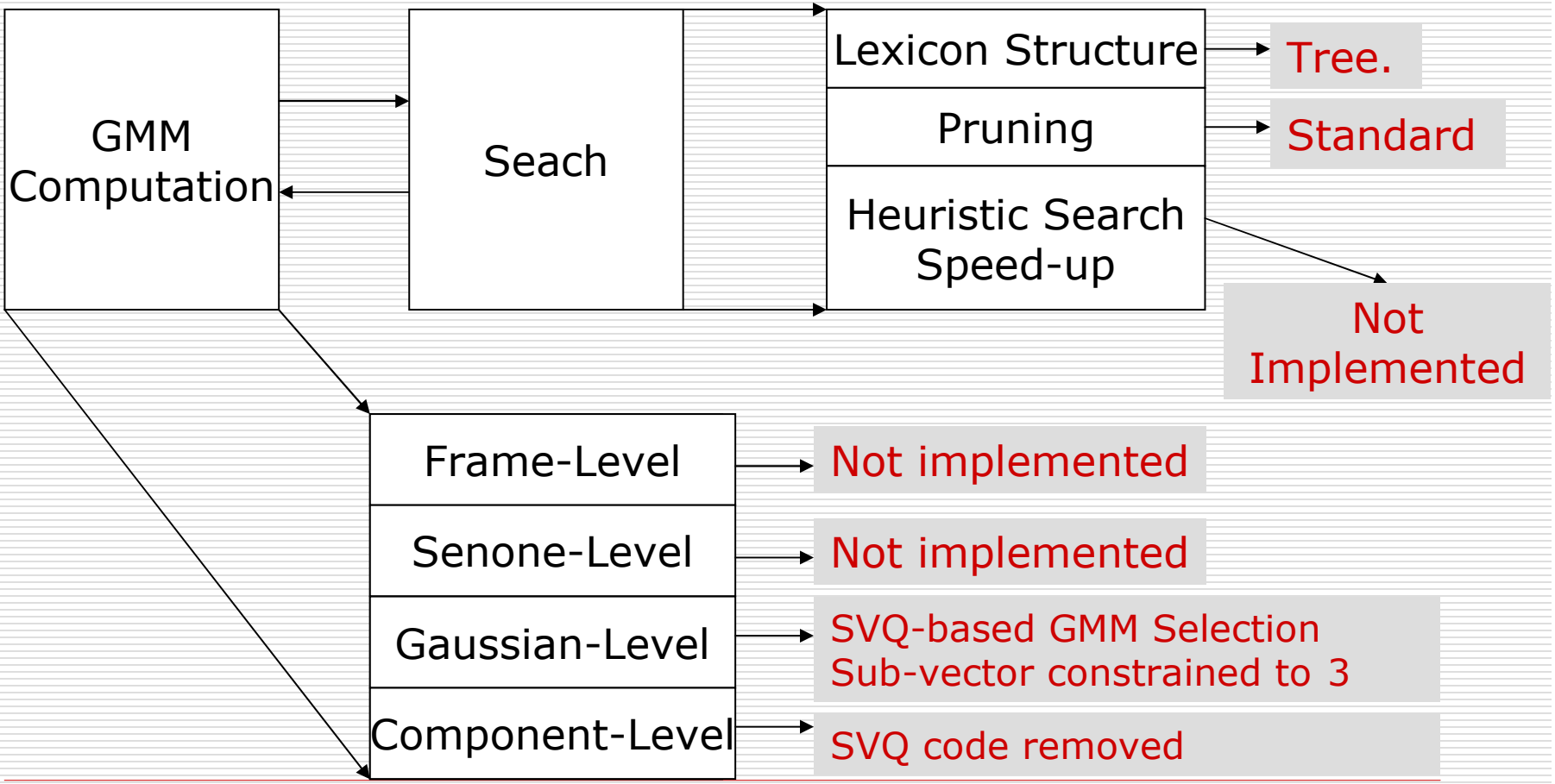
Summary of works in GMM Computation:

- 4-level of algorithmic optimization.
- However 2x2 !=4
  - There is a certain lower limit of computation (e.g. 75-80%)

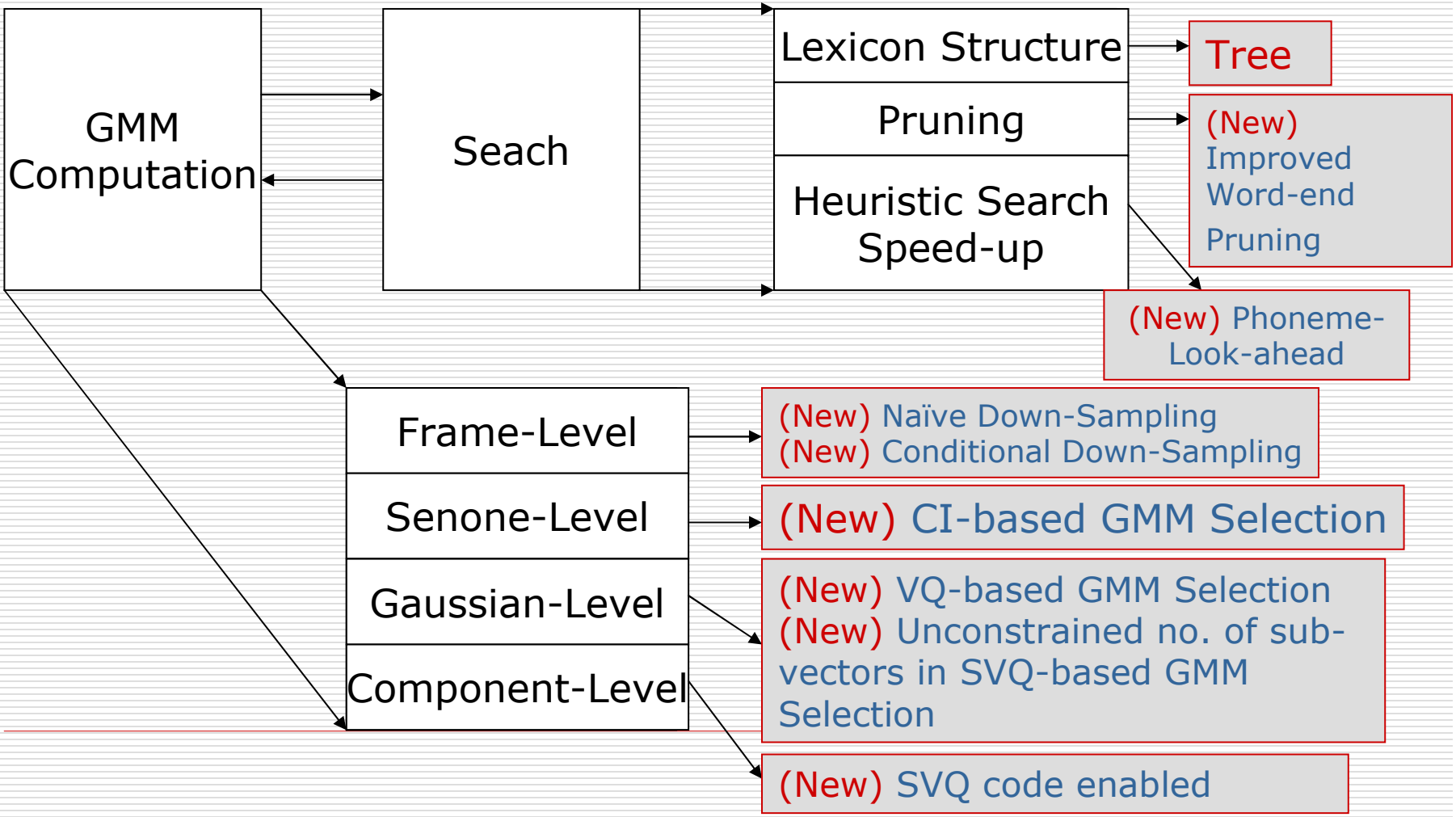
Work in improving search: Phoneme Look-ahead

- Phoneme Look-ahead
  - Use approximate senone scores of future frames to determine whether a phone arc should be extended.
  - Current Algorithm
    - If any senone of a phone HMM is active in any of future N frame, the phone is active.
    - Similar to Sphinx II.
- Results not very promising
  - Next step: try to add path-score in decision.

Speed-up Facilities in s3.3



Summary of Speed-up Facilities in s3.4



Language Model Facilities

- S3 and S3.3
  - Only accept non-class-based LM in DMP format.
  - Only one LM can be specified for the whole test set.
- S3.4
  - Basic facilities for accepting class-based LM in DMP format
  - Support dynamic LM
- Not yet thoroughly tested, may disable it before stable.

Availability

- Internal release to CMU initially
  - Put in Arthur's web page next week.
  - Include
    - speed-up code
    - LM facilities(?)
- If it is more stable, will put in Sourceforge.

Sphinx 3.5?

- Better interfaces
  - Stream-lined recognizer
- Enable Sphinx 3 to learn (AM and LM adaptation)
- Further Speed-up and improved accuracy
  - Improved lexical tree search
  - Machine optimization
  - Multiple recognizer combination?
- Your ideas:

Your help is appreciated.

- Current team:
  - Arthur =
    - (Maintainer + Developer) * Regression Tester ^ (Support)
  - Jahanzeb = Developer in Search+ Regression Tester
  - Ravi = Developer + Consultant
- We need,
  - Developers
  - Regression testers
  - Test scenarios
  - Extension of current code.
  - Suggestions
  - Comments/Feedbacks.
  - Talk to Alex if you are interested.