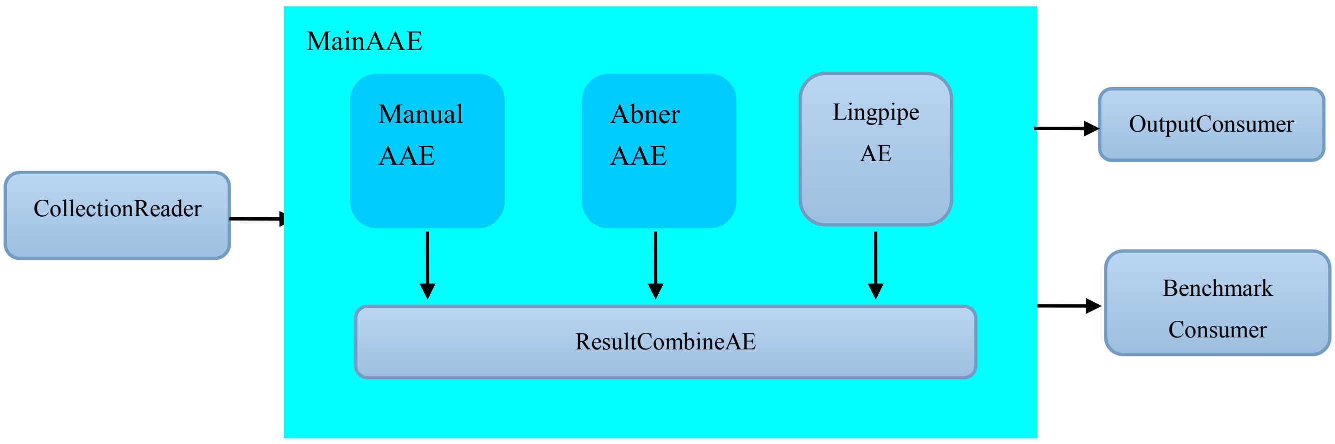
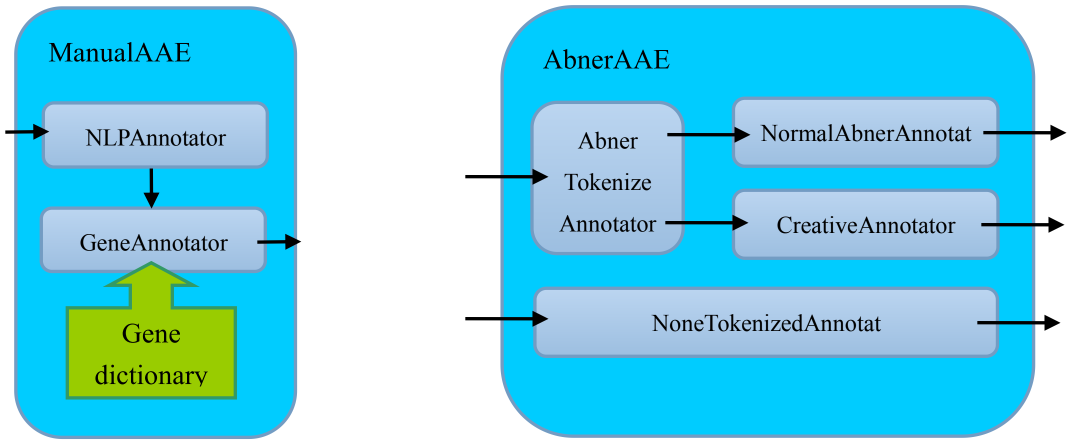
Report for Homework2: Xingyu Yan(xingyuy@andrew.cmu.edu)

Note:The system of hw2 is based on hw1, although did a lot of modification, it's basic structure remained the same. So I'll mainly write about those modifications rather than state all components once more.

1) System architect:





1) Inherited from HW1: Collection Reader, Manual AEE(where just called AAE in HW1), OutputConsumer and BenchmarkComsumer only made some little changes for new data structure.

2) Main AEE: Because I designed a aggregate analysis engine nesting structure, this is the main AAE to control the basic work flow of all analysis engines.

3) Lingpipe AE: Used Lingpipe as an answer system.

4) Abner AAE: I implemented Abner with 3 different configurations. Because tokenization of Abner is the most slow part and I got 2 AE needs tokenized result, I generated the tokenize part as a single AE, which significantly shortened the running time.

5) ResultCombine AE: I got 3 part of result, each part may have different idea about how to extract data. This AE is made for combine those extraction and provide a final result.

2) Outsource Using:

a) Manual AE:

i. ftp://ftp.ncbi.nih.gov/gene/DATA/GENE_INFO/All_Data.gene_info.gz Gene database;

    ii. Stanford NLP tool

- b) Abner AAE:
    - i. Abner v1.5
- c) Lingpipe AE:
    - i. Lingpipe v4.1.0, HMM model.

3) Problems when implementing:

- a) Abner's tokenize crises:

  Abner is a good tool of extracting gene name, but it have a really strange characteristics: when you are using it's tokenize tagger, the result will be tokenized as well instead of being the original. Even though that's really not so common, it is disturbing the location extracting for the words, which it failed to provide as well.

  In order to fix that, I coded a special class to change the recover the tokenized String to the original one, and search location/recover gene name at the same time.

- b) .Combination problem:

  Originally, my plan for the combination function is just vote: for the same area, 2 or more vote can be considered assured, and decide the shorter one be the final output if their length is not identical. But when I tested it, although it got a really high precision, it's recall is only about 1%, which is totally unacceptable. And when I looked at different combination, it's really embarrassing that Lingpipe did the best work on it's own. (If you want to see the result of the original combination, just config combination descriptor pointing to ResultCombineAnnotator_Original)

  That's really not a good sign because in that case, any add-on will damage the output. So finally my method is just "avoid Lingpipe". When any other method overlapped with Lingpipe, I give it away and only add extra tag when lingpipe did not cover.