Building Information Systems Automatically: Configuration Space Exploration for Biomedical Information Systems

Zi Yang¹, Elmer Garduno², Yan Fang³, Avner Maiberg¹, Collin McCormack⁴, Eric Nyberg¹

¹) Carnegie Mellon University {ziy, amaiberg, ehn}@cs.cmu.edu
²) Sinnia elmerg@sinnia.com
³) Oracle Corporation yan.fang@oracle.com
⁴) The Boeing Company collin.w.mccormack@boeing.com

10/30/2013 at CIKM 2013
Outline

• Introduction
• Related Work
• Problem Definition
• Modeling & Solution
• Open Source Implementation
• CSE For TREC Genomics QA
• Conclusion
Information Systems

• An information system

  – E.g. Information extraction systems, information retrieval systems, etc., which can also be represented as pipelines.

  – Watson: over 20 developers, over 5 years
Information Systems

• An information system
  – E.g. Information extraction systems, information retrieval systems, etc., which can also be represented as pipelines.
  – Watson: over 20 developers, over 5 years

How do we conventionally build an information system like this?
Conventional Information System Building Process
Conventional Information System Building Process
Conventional Information System Building Process
Conventional Information System
Building Process
Conventional Information System Building Process
Conventional Information System
Building Process
Conventional Information System Building Process
Conventional Information System Building Process
Is there a meta-system that can build an information system automatically with less human interference?
Automatically Building an Information System by Another Meta-System?
CSE Framework Overview

Configuration space exploration framework

Information processing task

Configuration space specification

Component characteristic modeling

Dynamic configuration selection

Optimal information system

Component pool

Algorithms

Toolkits

Knowledge bases

Benchmarks
The benefit of CSE framework
• Accelerate the system development cycle by automating the component selection and tuning!
• Save cost!

It requires
• Identify the tool, knowledge base, task algorithm candidates
• Provide information needs with known outcomes, e.g. answers to classical questions in the area.

Configuration space exploration framework
- Configuration space specification
- Component characteristic modeling
- Dynamic configuration selection

Information processing task
Optimal information system

Component pool
- Algorithms
- Toolkits
- Knowledge bases
- Benchmarks
Outline

• Introduction
• Related Work
• Problem Definition
• Modeling & Solution
• Open Source Implementation
• CSE For TREC Genomics QA
• Conclusion
Related Work

• Retrieval evaluation framework
  – Competitions and evaluation conferences: TREC, NTCIR, CLEF, etc.
  – Shared tasks that employ standardized data flows from successive subtasks: NTCIR ACLIA task (IR4QA and CCLQA), etc.
  – Workshop on Data infrastructurEs for Supporting Information Retrieval Evaluation

• Biomedical information retrieval and question answering
  – TREC Genomics, QA4MRE, BioASQ, etc.
Outline

• Introduction
• Related Work
• **Problem Definition**
• Modeling & Solution
• Open Source Implementation
• CSE For TREC Genomics QA
• Conclusion
Definition: Phase

• An information system
• A typical QA system:
  – Question analyzer, document retriever, passage extractor, answer generator

Phase $t$
The processing unit as the $t$-th step in a process
Definition: Component, Configuration

- Inside phase $t$

Component $f^c_t$
- An instantiated processing unit in phase $t$

Configuration $\omega^c_t$
- Consists of a set of parameters $\{\omega^{c,p}_t\}_p$

Configured component $f^c_t|\omega^c_t$
Definition: Component, Configuration

- In phase $t$ Named Entity Recognition in Questions:
  - Component $f_t^1$: a rule based NER
  - Component $f_t^2$: a CRF-based NER
  - Component $f_t^3$: an NER by knowledge base lookup
  - Configuration $\omega_t^1$: the set of rules
  - Configuration $\omega_t^2$: the weight vector trained for the model
  - Configuration $\omega_t^3$: knowledge base wrapper
Definition: Trace

A trace in QA

- Query tokenized by white-space string splitter, document retrieved from Indri repository index with default parameters, sentence extracted based on LingPipe sentence segmenter and VSM (Vector Space Model) similarity calculator

Trace $f^c|\omega^c = (f_1^{c_1}|\omega_1^{c_1}, f_2^{c_2}|\omega_2^{c_2}, ..., f_n^{c_n}|\omega_n^{c_n})$

An execution path that involves a single configured component for each phase
Definition: Trace

The number of traces grows exponentially as the number of phases or the number components increases.
Definition: Trace

Phase 1

Phase 2

Phase 3

Pipeline

Configuration space
\( F|\Omega = \{f^c|\omega^c\}_c \)

Set of all configured components
Definition: Cost

- Two characteristics of a configured component $f^c_t | \omega^c_t$

- **Cost** $c(f^c_t | \omega^c_t, x)$
  - How much resource is required to execute the component on input $x$?
  - E.g. execution time, storage space, network bandwidth, etc., measured by CPU time, allocated memory size, and data transfers.
  - The cost to execute the component on AWS.
Definition: Benefit

- Benefit $b(f_t^c | \omega_t^c, x)$
  - how much performance improvement can be achieved by executing it?
  - System level measurements include F-1, MAP, etc.
  - The benefit of a single component is difficult to measure without being integrated into a system with other components. How?
  - We assume it shares the benefit with the component that follows.
Definition: Cost, Benefit

- **Cost and benefit for a trace**

\[
c(f^c|\omega^c, x) = \sum_{t=1}^{n} c\left(f_t^{c_t}|\omega_t^{c_t}, x(c_1, \ldots, c_{t-1})\right)
\]

Sum of costs

\[
b(f^c|\omega^c, x) = b\left(f_n^{c_n}|\omega_n^{c_n}, x(c_1, \ldots, c_{n-1})\right)
\]

Final output from the last execution
Definition: Cost, Benefit

- **Cost and benefit for a configuration space**

  - \( c(\mathcal{F}|\Omega, x) = \sum_{t=1}^{n} \sum_{c_1=1}^{m_1} \ldots \sum_{c_t=1}^{m_t} c\left( f^{c_t}_t | \omega^{c_t}_t, x(c_1, ..., c_{t-1}) \right) \)

  
  Sum of unique executions

  All outputs from previous phases

  - \( b(\mathcal{F}|\Omega, x) = \max_{f^c|\omega^c \in \mathcal{F}|\Omega} b(f^c | \omega^c, x) \)

  Benefit of the best-performing trace

  - We can see \( c(\mathcal{F}|\Omega, x) \neq \sum_{f^c|\omega^c \in \mathcal{F}|\Omega} c(f^c | \omega^c, x) \)

  No repeated executions!
CSE Problem

• Configuration space exploration

\(m_t\) configured components for each of \(n\) phases:
\(f^1_t | \omega^1_t, f^2_t | \omega^2_t, ..., f^m_t | \omega^m_t\)

Input set \(S\)

Total resource capacity \(C\)

Optimal trace \(f^k | \omega^k\)
CSE Problem

• Constrained optimization problem

\[
\max_{F|\Omega \subseteq \mathcal{F}|\Omega, X \subseteq S} \mathbb{E}_x[b(F|\Omega, x)], \text{ s.t. } c(F|\Omega, X) \leq C
\]

- Select traces to execute
- Select inputs to test
- Random variable $x$, to ensure the generalizability to unseen data
- Total cost on the selected traces and inputs
CSE Problem

• How difficult is the problem?
  – *Trivial* if the total cost to execute all the configured components $\mathcal{F}|\Omega$ for all inputs $S$ does not exceed the capacity $C$.
  – *Nontrivial* for the real world problems.
  – For 12 components, with up to 4 parameters per component, and up to 6 options for parameter, there would be an estimated $6.050 \times 10^{13}$ executions!
Outline

• Introduction
• Related Work
• Problem Definition
• **Modeling & Solution**
• Open Source Implementation
• CSE For TREC Genomics QA
• Conclusion
Simplified CSE Problem

• CSE problem: a constrained optimization with variables (cost and benefit) unknown.

• A simplified CSE problem
  – The cost $c(f_t^C | \omega_t^C, x)$ of each configured component on each input is known;
  – The benefit $b(f_t^C | \omega_t^C, x)$ is i.i.d. for each $f_t^C | \omega_t^C$
Simplified CSE Problem

• Solution for the simplified CSE problem
  – Adding configured components and inputs in descending order of least cumulative cost.

\[ \text{LCC}(f_t^c|\omega_t^c, x) = c(f_t^c|\omega_t^c, x) + \sum_{u=t+1}^{n} \min_{c_u} c(f_u^c u|\omega_u^c, x) \]

Original cost  Sum of min costs on the suffix, to get evaluated ASAP!
Basic Solution Ideas

• Relax the first assumption
  – Stochastic scheduling problem (or stochastic knapsack problem)

**Deterministic Knapsack Problem**
- Known constant cost and benefit for each element

**Stochastic Knapsack Problem**
- Random variables with known probability distribution
Basic Solution Ideas

• Relax the second assumption. Why not i.i.d.?
  – A weak component tends to exhibit low performance regardless of input $x$;
  – A complex $x$ may lower the performance regardless of the component.

• To better estimate the distributions... hierarchical Bayesian modeling
Basic Solution Ideas

• Dependency of characteristics
  – E.g., high-performing components tend to achieve again high performance combined with other components.
  – E.g., low-performing components tend to be pruned from the pool.

• Prior knowledge
  – E.g., retrieving relevant passage requires more resources than tokenizing question texts.
Solution

- Modeling cost and benefit distributions

  - Global hyper-parameters $\gamma$ and $\beta$
  - Phase-level parameters $\gamma_t$ and $\beta_t$
  - Configured component level parameters $\gamma^c_t$ and $\beta^c_t$
  - Observed $c(f^c_t | \omega^c_t, x)$ and $b(f^c_t | \omega^c_t, x)$
  - Input parameters $\gamma^x$ and $\beta^x$
Solution

• CSE as a stochastic scheduling problem

\[
\max_{F|\Omega \subseteq \mathcal{F}|\Omega, X \subseteq S} \mathbb{E}_b[b(F|\Omega, X)], \text{ s.t. } \mathbb{E}_c[c(F|\Omega, X)] \leq C
\]

Both cost and benefit are random variables

• In most general cases, no efficient optimal solution

• A greedy algorithm with *adaptive policies*
  – Dynamically choose which traces to execute next based on previous executed traces
Solution

• Greedy algorithm for CSE problem

Check capacity $C$

Predict $c(f_t^C|\omega_t^C, x)$ and $b(f_t^C|\omega_t^C, x)$

Find the $f_t^C|\omega_t^C, x$ pair to maximize $h$

Execute traces involving $f_t^C|\omega_t^C$ on $x$

Update parameters $\gamma_t^C, \beta_t^C, \gamma^x, \beta^x$

Examples of $h$

**Cost relevant**

$-E_c[LCC(f_t^C|\omega_t^C, x)]$

**Benefit relevant**

$E_b[b(f_t^C|\omega_t^C, x)]$

**Benefit density**

$E_b[b(f_t^C|\omega_t^C, x)] / E_c[LCC(f_t^C|\omega_t^C, x)]$

**Profit**

$E_b[b(f_t^C|\omega_t^C, x)] - \lambda E_c[LCC(f_t^C|\omega_t^C, x)]$
Example: 3-stage Linear Characteristics

- 3-stage linear characteristics

Global hyper-parameters $\gamma$ and $\beta$

Phase-level parameters $\gamma_t$ and $\beta_t$

Configured component level parameters $\gamma_t^c$ and $\beta_t^c$

Input parameters $\gamma^x$ and $\beta^x$

\[ \gamma_t \sim \mathcal{N} (\gamma, \tau^2) \]
\[ \beta_t \sim \mathcal{N} (\beta, \sigma^2) \]

\[ \gamma_t^c \sim \mathcal{N} (\gamma_t, \tau_t^2) \]
\[ \beta_t^c \sim \mathcal{N} (\beta_t, \sigma_t^2) \]

\[ c(f_t^c | \omega_t^c, x) \sim \mathcal{N} (\gamma_t^c, \tau_t^{c2}) \]
\[ b(f_t^c | \omega_t^c, x) \sim \mathcal{N} (\beta_t^c, \sigma_t^{c2}) \]
Example: 3-stage Linear Characteristics

• Parameter estimation
  – Posterior distributions for $\gamma_t^c, \beta_t^c, \gamma_t, \beta_t$ can be easily derived, in the same normal form $\mathcal{N}(\phi^{-1}\theta, \phi^{-1})$.
  – Details omitted
Bayesian Parameter Estimation

- At the beginning of the experiment
  - When \( n_t^c = 0 \) for all \( t \) and \( c \), each parameter takes the same value from the user specified hyperparameter (\( \gamma \) and \( \beta \)).

<table>
<thead>
<tr>
<th>Global hyper- parameter ( \gamma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1 parameter ( \gamma_1 )</td>
</tr>
<tr>
<td>Phase 2 parameter ( \gamma_2 )</td>
</tr>
</tbody>
</table>
Bayesian Parameter Estimation

• As the experiment proceeds
  – Each parameter is dynamically shifted way to better model the execution behavior of the phase or component (the average of past execution records).

<table>
<thead>
<tr>
<th>Global hyper-parameter $\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1 parameter $\gamma_1$</td>
</tr>
<tr>
<td>Phase 2 parameter $\gamma_2$</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Related Work
- Problem Definition
- Modeling & Solution
- Open Source Implementation
- CSE For TREC Genomics QA
- Conclusion
Architecture & Highlights

- Project GitHub Page: [http://oaqa.github.io](http://oaqa.github.io)

**Dependency Frameworks**
- JAVA
- UIMA
  - To date the only industry standard for content analytic

**Our Abstract Frameworks**
- CSE
  - Provide abstract components and drivers to help easily set up experiments for UIMA pipelines.
- BaseQA
  - Contains abstract components for UIMA-based QA pipelines.
Architecture & Highlights

- Project GitHub Page: [http://oaqa.github.io](http://oaqa.github.io)

**Frameworks**
- Our Abstract Frameworks
  - Provide abstract components and drivers to help easily set up experiments for UIMA pipelines.
  - Contains abstract components for UIMA-based QA pipelines.

**Applications**
- Implement drivers and connectors for specific experiment environments
  - Wrap resources and tools
  - Implement QA algorithms and pipeline logics

**UIMA**
- JAVA
  - To date the only industry standard for content analytic

**CSE**
- Automatic data persistence
- Configurable evaluation
- Declarative descriptors

**BaseQA**
- Graphical user interface
- Distributed architecture

**Dependency Frameworks**
- Architecture & Highlights
  - Distributed architecture
  - Automatic data persistence
  - Configurable evaluation
  - Declarative descriptors
  - Graphical user interface
Our Abstract Frameworks

- Contains abstract components for UIMA-based QA pipelines

Our Applications, components

- BaseQA
- BioQA
- resource-wrappers
- indri-provider
- solr-provider

Implement QA algorithms and pipeline logics
Implements drivers and connectors for specific experiment environments
Wrap resources and tools

Global resource caching
Outline

• Introduction
• Related Work
• Problem Definition
• Modeling & Solution
• Open Source Implementation
• CSE For TREC Genomics QA
• Conclusion
• Basic abstract pipeline is specified in BaseQA, which is designed to include three major pipeline phases.
TREC Genomics

• Retrieval of short passages that specifically address an information need expressed as a question (64 in total).
  – E.g., what [MOLECULAR FUNCTIONS] is LITAF involved in?

• Corpus: > 100K full text articles in HTML format

• Evaluation: DocMAP, PsgMAP
# Resource and Algorithm Integration

<table>
<thead>
<tr>
<th>Category</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP tools</td>
<td>LingPipe HMM based tokenizer</td>
</tr>
<tr>
<td></td>
<td>LingPipe HMM based POS tagger</td>
</tr>
<tr>
<td></td>
<td>LingPipe HMM based named entity recognizer</td>
</tr>
<tr>
<td></td>
<td>Rule based lexical variant generator</td>
</tr>
<tr>
<td>Knowledge bases</td>
<td>UMLS as syn-/acronym resolution</td>
</tr>
<tr>
<td></td>
<td>EntrezGene as syn-/acronym resolution</td>
</tr>
<tr>
<td></td>
<td>MeSH as syn-/acronym resolution</td>
</tr>
<tr>
<td>Retrieval tools</td>
<td>Indri system indexed for the full texts</td>
</tr>
<tr>
<td>Reranking algorithms</td>
<td>Important sentence identification</td>
</tr>
<tr>
<td></td>
<td>Term proximity based ranking</td>
</tr>
<tr>
<td></td>
<td>Score combination of different retrieval units</td>
</tr>
<tr>
<td></td>
<td>Overlapping passage resolution</td>
</tr>
</tbody>
</table>
BioQA for TREC Genomics

- Components are then assembled into BioQA pipeline
# Experiment Settings

- **CSE**: moderate configuration space
- **Scaled CSE**: larger configuration space
  - Same component and capacity
- Compared with TREC participating systems

<table>
<thead>
<tr>
<th>Component</th>
<th>Participants</th>
<th>CSE</th>
<th>Scaled CSE</th>
</tr>
</thead>
<tbody>
<tr>
<td># Component</td>
<td>~1,000</td>
<td>&gt;</td>
<td>=</td>
</tr>
<tr>
<td># Configuration</td>
<td>~1,000</td>
<td>&gt;</td>
<td>=</td>
</tr>
<tr>
<td># Trace</td>
<td>92</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td># Execution</td>
<td>~1,000</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>Capacity (hours)</td>
<td>N/A</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CSE helps identify a better system with fewer and more basic components.

24 hours allows Scaled CSE to identify a system that greatly outperforms the TREC median.
Best Baseline System

- **Knowledge bases**
  - Acronym expanded by UMLS
  - Synonym expanded by MeSH + EntrezGene
  - Lexical variants added

- **Term weighting**
  - Concept: 0.8, Gene: 0.4, Verb: 0.1, Regular: 0.1

- **Retrieval**
  - Indri, smoothed with Jelinek-mercer ($\lambda$: 0.1)
  - Combining reciprocal of rank of document and passage level retrieval scores, document score weighted by 0.1
Best Baseline System

• Sentence extraction
  – # Sentences in a passage: 1 at maximum
  – # Passages in a paragraph: 2 at maximum
  – Important sentence similarity: Overlap coefficient with threshold 0.3
  – Neighboring sentence similarity: matching coefficient with threshold 1

• Proximity-based reranking
  – C1: 1.0E-4, C2: 100 C3: 1.0E-5
  – Combining original score, weighted by 0.7

• Identical passages removed
Component analysis: KB

### DocMAP
- None
- UMLS
- MeSH
- EntrezGene
- UMLS + MeSH
- UMLS + EntrezGene
- UMLS + MeSH + EntrezGene

### PsgMAP
- None
- MeSH
- EntrezGene
- UMLS + MeSH
- UMLS + EntrezGene
- UMLS + MeSH + EntrezGene

Graphs showing the performance of synonym and acronym expansion with KB-based knowledge bases.
Component analysis: Score Combiner

**DocMAP**

- Exp of Norm
- Exp
- Log
- Norm of Exp
- Norm
- None

**PsgMAP**

- Exp of Norm
- Exp
- Log
- Norm of Exp
- Norm
- None

**DocMAP**

- Lexical variants
- Setence extraction
- Proximity reranking
- Filtering

**PsgMAP**

- Lexical variants
- Setence extraction
- Proximity reranking
- Filtering

Document/passage score combiner

Synonym Excluded
Component analysis: Score Combiner

Comparison generated automatically!
No human interference!
Repeatable!
Framework Performance Analysis

Cost estimation at the end

Phase-level

Component-level

Posterior Likelihood

Cost estimation at the beginning (1%)
Framework Performance Analysis

Cost estimation at the end

Phase-level

Component-level

Posterior

Likelihood

Distribution shifted from the prior over time

Posterior Likelihood

As experiment proceeds, more samples made the SD of posterior smaller

Cost estimation at the beginning (1%)
Framework Performance Analysis

Component-level benefit estimation at 20%

Likelihood

Posterior

Downstream sentence extraction and retrieval
- More frequently tested

Upstream KB based query expansion
- Less frequently tested
Generalizability Analysis

- Resampling methods based on generalizability for trace selection
- Test on a related but different unseen set (TREC Genomics 2007)
QA4MRE

- Question Answer for Machine Reading Evaluation at CLEF 2012/2013
  - Test document to read, multiple choice questions to answer
- Configuration space
  - 12 UIMA components were first developed
  - Replace UIMA descriptors with CSE Descriptor
- CSE
  - 46 configurations, 1,040 combinations, 1,322 executions
  - The best trace identified by CSE achieved 59.6% performance gain over the original pipeline!
- Alkesh Patel, Zi Yang, Eric Nyberg, Teruko Mitamura. Building an Optimal Question Answering System Automatically using Configuration Space Exploration (CSE) for QA4MRE 2013 Tasks
Outline

• Introduction
• Related Work
• Problem Definition
• Modeling & Solution
• Open Source Implementation
• CSE For TREC Genomics QA
• Conclusion
Conclusion

• Novel problem and solution
  – Configuration Space Exploration problem for rapid domain adaptation

• Open source implementation
  – CSE framework and other application projects

• Case study on building a biomedical QA system
  – Demonstrate the effectiveness of the approach
  – Provide insights on the BioQA components
Future Work

• To apply the framework to solve the CSE problems in other domains to benefit automatic system construction.
  – *Looking for collaborations!*

• To deal with continuous parameters.

• Automatic phase design.
Thanks

Project GitHub Page:  http://oaqa.github.io
Email me:  ziy@cs.cmu.edu

Zi Yang
Language Technology Institute
School of Computer Science
Carnegie Mellon University

10/30/2013 at CIKM 2013

Excluding pictures from the Web
## Experimental results

<table>
<thead>
<tr>
<th></th>
<th>Participants</th>
<th>CSE</th>
<th>Scaled CSE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># Component</strong></td>
<td>~1,000</td>
<td>12</td>
<td>= 12</td>
</tr>
<tr>
<td><strong># Configuration</strong></td>
<td>~1,000</td>
<td>32</td>
<td>&lt;= 2,946</td>
</tr>
<tr>
<td><strong># Trace</strong></td>
<td>92</td>
<td>2,700</td>
<td>&lt;= 1.426 × 10^{12}</td>
</tr>
<tr>
<td><strong># Execution</strong></td>
<td>~1,000</td>
<td>190,680</td>
<td>&lt;= 6.050 × 10^{13}</td>
</tr>
<tr>
<td><strong>Capacity (hours)</strong></td>
<td>N/A</td>
<td>24</td>
<td>= 24</td>
</tr>
<tr>
<td><strong>DocMAP</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>.5439</td>
<td>.5640</td>
<td>.5072</td>
</tr>
<tr>
<td>Median</td>
<td>.3083</td>
<td>.1770</td>
<td>.3509</td>
</tr>
<tr>
<td>Min</td>
<td>.0198</td>
<td>.1087</td>
<td>.2679</td>
</tr>
<tr>
<td><strong>PsgMAP</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>.1486</td>
<td>&lt;</td>
<td>.1181</td>
</tr>
<tr>
<td>Median</td>
<td>.0345</td>
<td>.1603</td>
<td>.0713</td>
</tr>
<tr>
<td>Min</td>
<td>.0007</td>
<td>.0311</td>
<td>.0164</td>
</tr>
</tbody>
</table>
Best Baseline system

1. DefaultKeytermExtractor
2. LingPipeHmmPosTagger[ModelFilePath:/pos-en-bio-medpost.HiddenMarkovModel]
3. LingPipeAndAbbreviation[ModelFilePath:/ne-en-biogenia.TokenShapeChunker]
4. AddLexicalVariants
5. KeytermRestorer[KeytermFilePath:/keyterm-cache/mesh-entrezgene-umls]
6. UmlsSynonymLookupper
7. KeytermRefiner[MESH:true#ENTREZ:true#UMLS:false#LexicalVariants:true#PosTagger:true#MESH-Acronym:false#ENTREZ-Acronym:false#UMLS-Acronym:true#concept-term-weight:0.8#regular-term-weight:0.1#verb-term-weight:0.4#gene-term-weight:0.4]
Best Baseline system

8 | IndriRetrievalStrategist[hit-list-size:1000#smoothing:jelinek-mercer#smoothing-mu:2500#smoothing-lambda:1.0]
9 | IndriLegalSpanPassageExtractor[hit-list-size:1000#PassageSpan:legalspan#smoothing:jelinek-mercer#smoothing-mu:2500#smoothing-lambda:0.1]
10 | ImportantSentenceExtractor[MaxNumSentInPassage:1#ConsiderSynonyms:true#MaxNumPassageInParagraph:2#ImportantSentSimThreshold:0.3#ImportantSentSim:similarity.ModifiedOverlapCoefficient#NeighborSentSimThreshold:1.0#NeighborSentSim:similarity.MatchingCoefficient]
11 | TermProximityRanker[C1:1.0E-4#C2:100#C3:1.0E-5#ConsiderSynonyms:true#CombineOriginalScores:true#OriginalWeight:0.7]
12 | ScoreCombiner[DocWeight:0.1#TransformMode:reciprocal_of_rank]
13 | OverlappingPassageFilter[limit:1000#FilterMode:FILTER_IDENTICAL]