The Use of Text Retrieval and Natural Language Processing in Software Engineering

Sonia Haiduc

Assistant Professor
Department of Computer Science
Florida State University

Sonia Haiduc

- Academic background:

B.Sc in Romania

MS, Ph.D in Detroit, USA

Florida State University

Tallahassee, FL

SERENE Lab

- **Sonia Haiduc**, Assistant Professor

- Ph.D. Students:

Javier Escobar Avila

Chris Mills

Esteban Parra Rodriguez

Main research interests

- Software maintenance and evolution
- Program comprehension
- Source code analysis
- Mining software repositories
- Developer performance and efficiency

Our goal

- Help software developers to build and maintain software faster and better
- We often leverage techniques from outside SE
  - Information Retrieval
  - Natural Language Processing
  - Machine Learning

The Use of Text Retrieval and Natural Language Processing in Software Engineering

Sonia Haiduc

Assistant Professor
Department of Computer Science
Florida State University

Textual Information in Software

- Captures concepts of the problem domain, developer intentions, developer communication, etc.

- Found in many software artifacts:
  - Requirements
  - Design documents
  - Source code (identifiers, comments)
  - Commit notes
  - Documentation
  - User manuals
  - Q/A websites (StackOverflow, etc.)
  - Developer communication (emails, chat, tweets, etc.)
  - …

Text Retrieval

- *Information Retrieval (IR)*: the process of actively seeking out information relevant to a topic of interest (*van Rijsbergen*)

- **Text Retrieval (TR):** a branch of IR where the information is in text format
  - Search space: collection of documents (*corpus*)
  - *Document* - generic term for an information unit
    - book, chapter, article, webpage, etc.
    - class, method, interface, etc.
    - individual requirement, bug description, test case, e-mail, design diagram, etc.

Natural Language Processing

- Refers to the use and ability of systems to process sentences in a natural language such as English (rather than in a specialized, artificial computer language such as C++)

- Combines techniques from computer science, artificial intelligence, computational linguistics, probability and statistics

TR and NLP in Software Engineering

- Applied to over 30 different SE tasks
  - Traceability Link Recovery
  - Feature/concept/concern/bug location
  - Code reuse
  - Bug triage
  - Program comprehension
  - Architecture/design recovery
  - Quality assessment and measurement
  - Software evolution analysis
  - Defect prediction and debugging
  - Automatic documentation
  - Testing
  - Requirements analysis
  - Restructuring/refactoring
  - Software categorization
  - Licensing analysis
  - Impact analysis
  - Clone detection
  - Effort prediction/estimation
  - Domain analysis
  - Web services discovery
  - Use case analysis
  - Team management, etc.

Using TR and NLP for Retrieving Software Artifacts

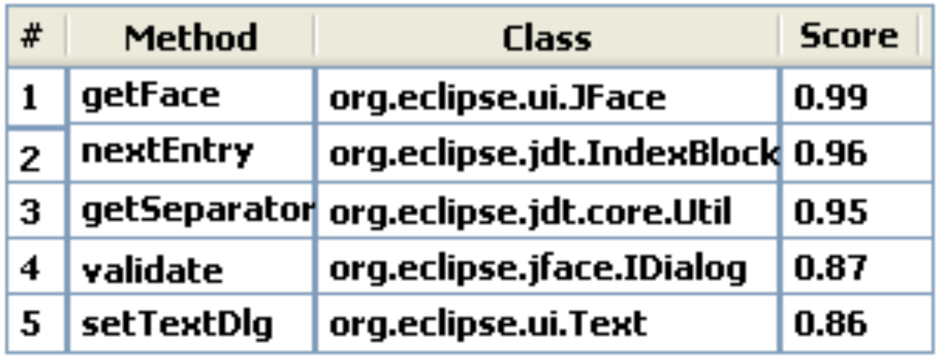
- Formulate the SE task as a retrieval problem and find the software artifacts that satisfy a particular information need

- Some examples:
  - *Bug Location*: retrieve all methods in the code relevant for a particular bug report;
  - *Bug Report De-duplication*: find all bug reports that already exist and are similar to a new bug report, in order to prevent duplication.
  - *Bug Triage*: given a new bug report, find the solved bug report that is most similar to the new one and assign it to the same developer.
  - *Feature Location*: find the classes in the code that implement a particular feature or concept;
  - *Code Reuse*: retrieve pieces of code or entire systems online that implement a particular functionality;
  - *Clone and plagiarism detection*: given a piece of code (e.g., a method), find similar pieces of code to it and mark them as potential clones.
  - *Defect Prediction*: given a method or class, estimate the number of it contains by extrapolating from similar artifacts for which the number of defects is known.
  - *Impact Analysis*: when changing a method, determine other methods that may be impacted by the change by finding the similar methods to it.

Retrieving Relevant Software Artifacts

Query

INPUT



#	Method	Class	Score
1	getFace	org.eclipse.ui.JFace	0.99
2	nextEntry	org.eclipse.jdt.IndexBlock	0.96
3	getSeparator	org.eclipse.jdt.core.Util	0.95
4	validate	org.eclipse.jface.IDialog	0.87
5	setTextDlg	org.eclipse.ui.Text	0.86

Relevant Artifacts

TR/NLP Model

```
    OUTPUT_BYTE(h);
    OUTPUT_BYTE(w>>BYTE_SIZE);
    OUTPUT_BYTE(w);

    order1 = order>>4;
    order2 = order & 15;

#ifdef TRACE
    if (!(fpm = fopen("ppmenc.doc", "wb"
    {
        fprintf(stderr, " \n Error: Ca
        exit(2);
    }
#endif
        /* allocate 'order+1' element
```

Software Artifacts

Steps

1. Create and preprocess corpus using light NLP
2. Index corpus – choose the *TR model*
3. Formulate a *query*
    - Manual or automatic
4. Compute *similarities* between the query and the documents in the corpus (i.e., relevance)

5. *Rank* the documents based on the similarities

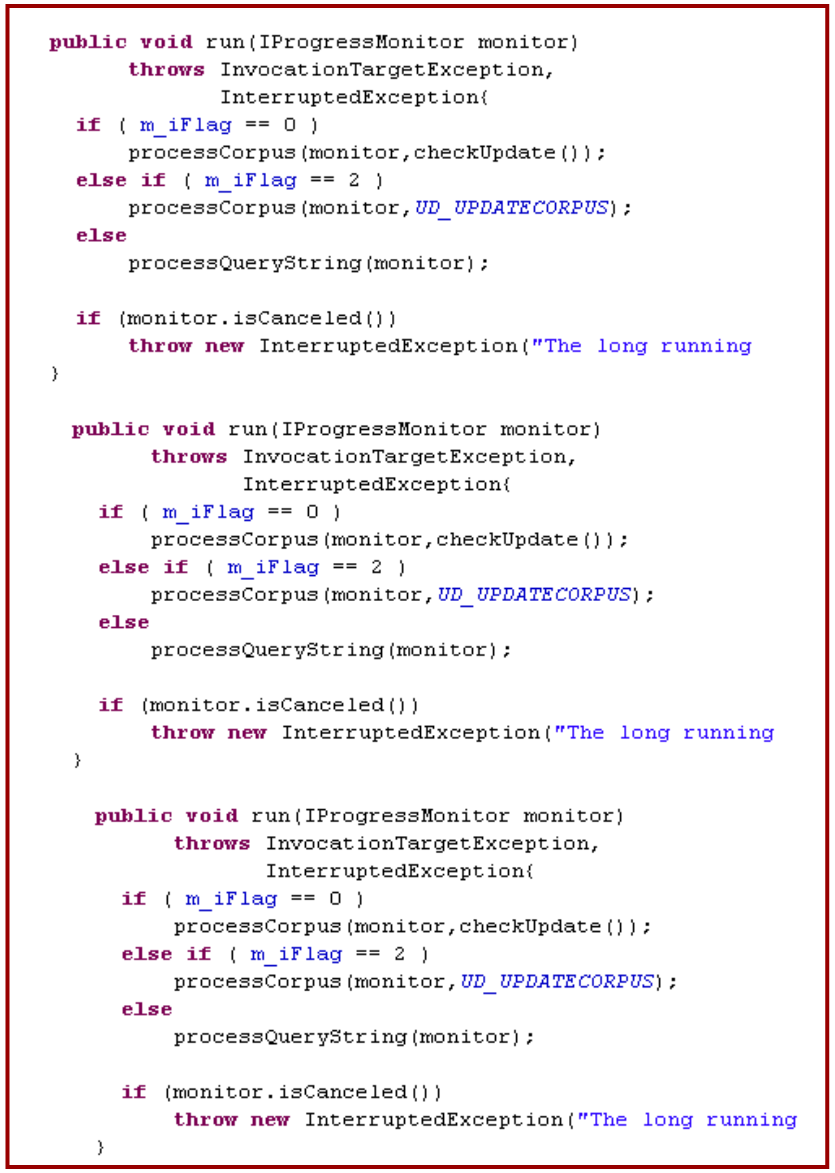
6. Return the top N documents as the *result list*
7. Inspect the results
8. GO TO 3. if needed or STOP

Creating and Preprocessing a Software Corpus

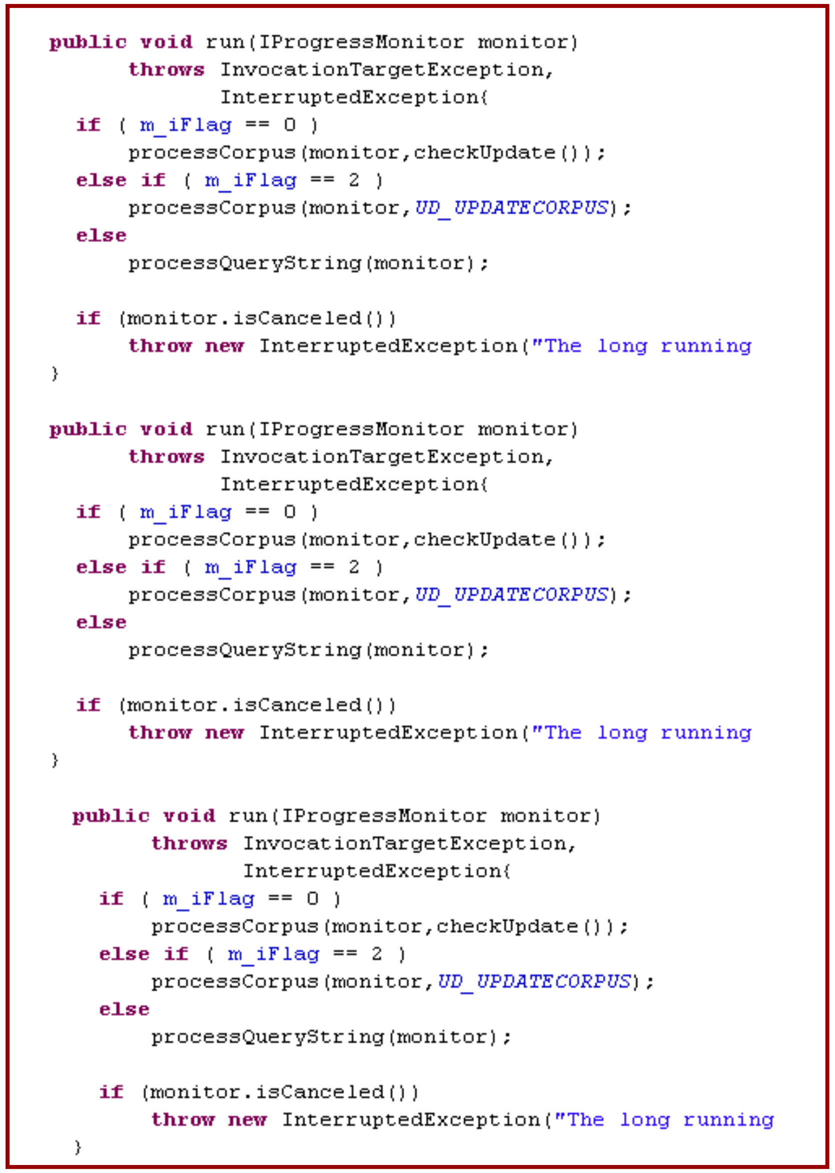
- Parsing software artifacts and extracting documents
  - ***corpus*** = a collection of documents (e.g., methods)
- Text normalization (white space and non-textual tokens removal, tokenization)
- Splitting: split_identifiers, SplitIdentifiers, etc.
- Stop words removal
  - common words in English, programming language keywords, project-specific words, etc.
- Abbreviation and acronym expansion
- Stemming

Extracting Documents

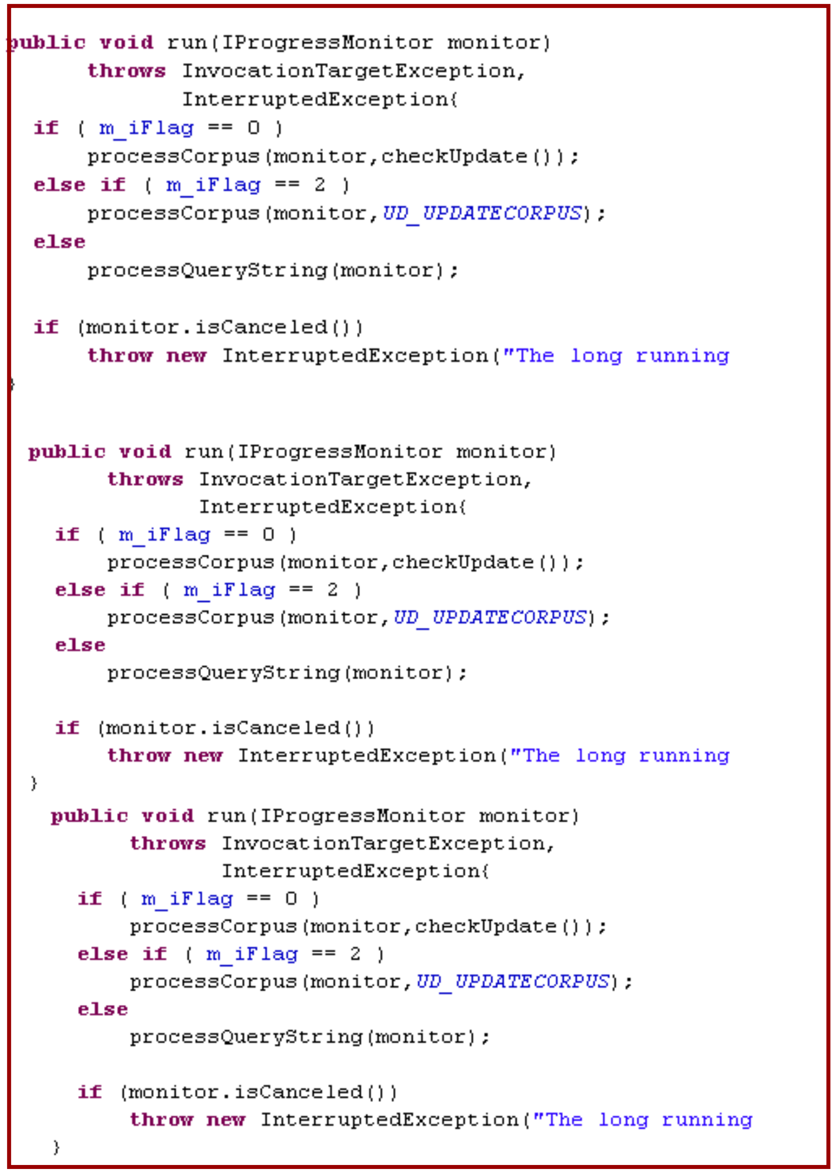
- Documents can be of different types and granularities (e.g., methods, classes, files, emails, paragraphs, bug descriptions, etc.)



```
public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}

public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}

public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}
```



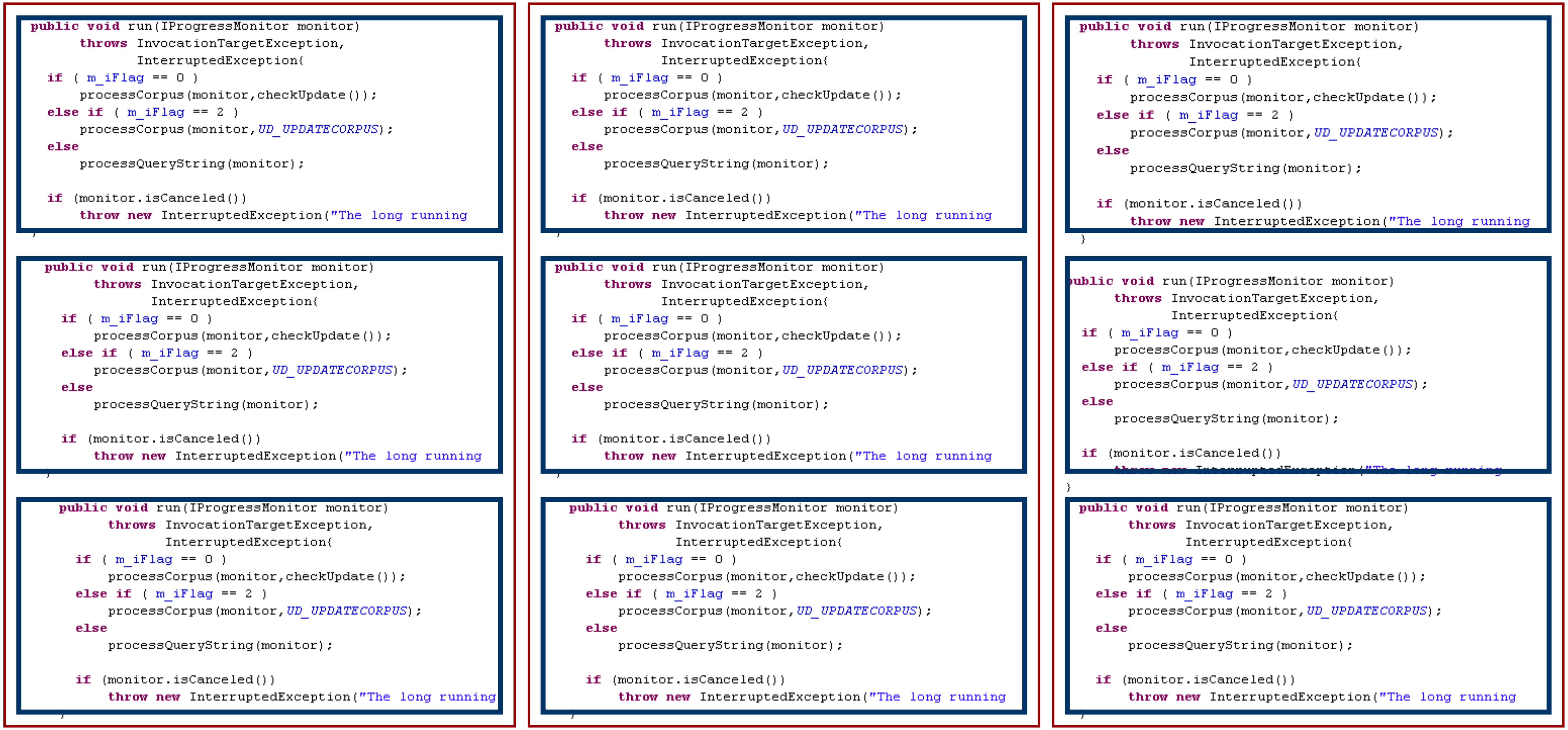
```
public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}

public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}

public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}
```



```
public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}

public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}

public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);

  if (monitor.isCanceled())
      throw new InterruptedException("The long running
}
```

Extracting Documents

- Documents can be of different types and granularities (e.g., methods, classes, files, emails, paragraphs, bug descriptions, etc.)



```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
```

```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
```

```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
```

```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
```

```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
```

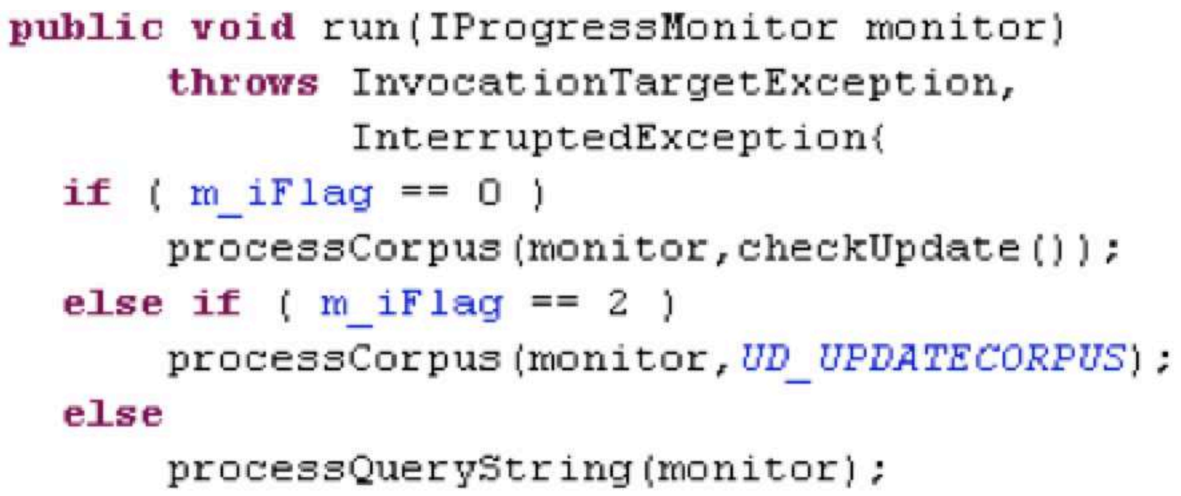
```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
```

```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
}
```

```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
}
```

```java
public void run(IProgressMonitor monitor)
        throws InvocationTargetException,
                  InterruptedException{
   if ( m_iFlag == 0 )
        processCorpus(monitor,checkUpdate());
   else if ( m_iFlag == 2 )
        processCorpus(monitor,UD_UPDATECORPUS);
   else
        processQueryString(monitor);

   if (monitor.isCanceled())
        throw new InterruptedException("The long running
```

Transform Source Code to Plain Text



```
public void run(IProgressMonitor monitor)
      throws InvocationTargetException,
             InterruptedException{
  if ( m_iFlag == 0 )
      processCorpus(monitor,checkUpdate());
  else if ( m_iFlag == 2 )
      processCorpus(monitor,UD_UPDATECORPUS);
  else
      processQueryString(monitor);
```

public void run IProgressMonitor monitor throws
InvocationTargetException InterruptedException if m_iFlag
processCorpus monitor checkUpdate else if m_iFlag
processCorpus monitor UD_UPDATECORPUS else
processQueryString monitor

Text Normalization

- Remove whitespace and non-textual characters
- Break up the text in meaningful “tokens” and keep only what makes sense

- Pay attention to:
  - Numbers: “P450”, “2001”
  - Hyphenation: “MS-DOS”, “R2-D2”
  - Punctuation: “John’s”, “command.com”
  - Case: “us”, “US”
  - Phrases: “venetian blind”

Splitting

- Splitting: decomposing identifiers into their compound words
- Identifiers may use division markers (e.g., camelCase and _), or may not
- Examples:
  - `getName` -> ‘get’, ‘Name’
  - `getMAXstring` -> ‘get’, ‘MAX’, ‘string’
  - `ASTNode` -> ‘AST’, ‘Node’
  - `account_number` -> ‘account’, ‘number’
  - `simpletypename` -> ‘simple’, ‘type’, ‘name’

Stop Words

- Very frequent words, with no power of discrimination (e.g., programming language keywords, common English terms)
- Typically function words, not indicative of content (e.g., “the”, “class”)
- The stop words set depends on the document collection and on the application

Abbreviation and Acronym Expansion

- Expand abbreviations and acronyms to the corresponding full words

- Examples:
  - `mess` -> ‘message’
  - `src` -> ‘source’
  - `auth` -> ‘authenticate’ OR ‘author’?

Stemming

- Identify morphological variants, creating “classes”
  - system, systems
  - forget, forgetting, forgetful
  - analyse, analysis, analytical, analysing

- Replace each term by the class representative (root or most common variant)

TR and NLP Models

- The TR/NLP model indexes the information in the corpus for fast retrieval
- Different TR/NLP models represent the same corpus differently and can lead to different search results
- Most Popular TR and NLP Models Used in SE:
  - Vector Space Model (VSM)
  - Latent Semantic Indexing (LSI)
  - Latent Dirichlet Allocation (LDA)
  - Okapi BM25 and BM25F
  - Language Models

Query Formulation

- A query is formulated that captures the information need of the developer
  - can be manually formulated by the developer (e.g., “copy paste” – for finding the classes that implement the copy-paste feature in an editor)
  - can be automatically formulated based on a software artifact or written information need (e.g., extract a query directly from a bug report written by a user or another developer)
- The query is then preprocessed using the same approach used on the corpus

Simple Query Improvements

- Spellchecking -> correct words

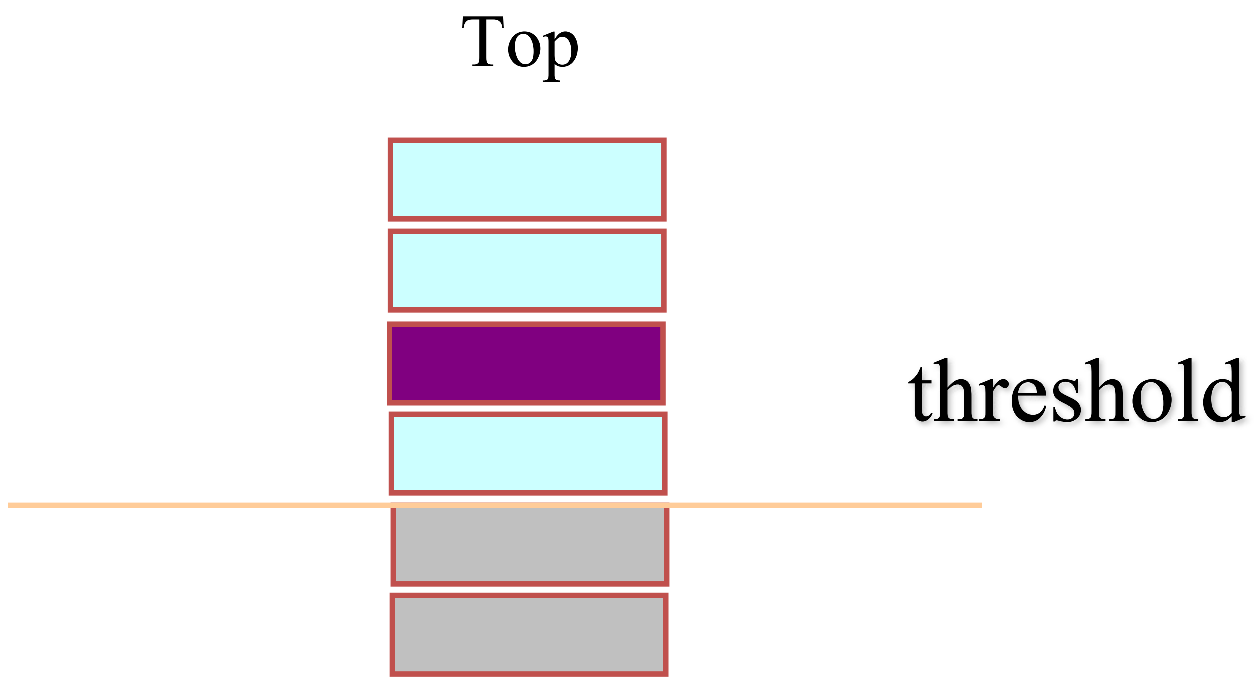
- Compare with software vocabulary
  - remove words that do not appear in the software system
  - use software thesaurus to suggest alternative words (i.e., synonyms)

Query Reformulation

- How can we reformulate a bad query?
  - ***Thesaurus expansion***:
    - Suggest terms similar to query terms
  - ***Relevance feedback***:
    - Suggest terms (and documents) similar to retrieved documents that have been judged to be relevant
  - More advanced: automatic based on query properties, mining terms from source code, etc.

Evaluation

- For a given query, produce the ranked list of documents.
- Determine a threshold and cut the ranked list such that only the results up to the threshold are considered as retrieved.



- Mark each document in the top results (up to the threshold) that is relevant according to the gold standard.
- Note: different thresholds on the ranked list produces different sets of retrieved documents.

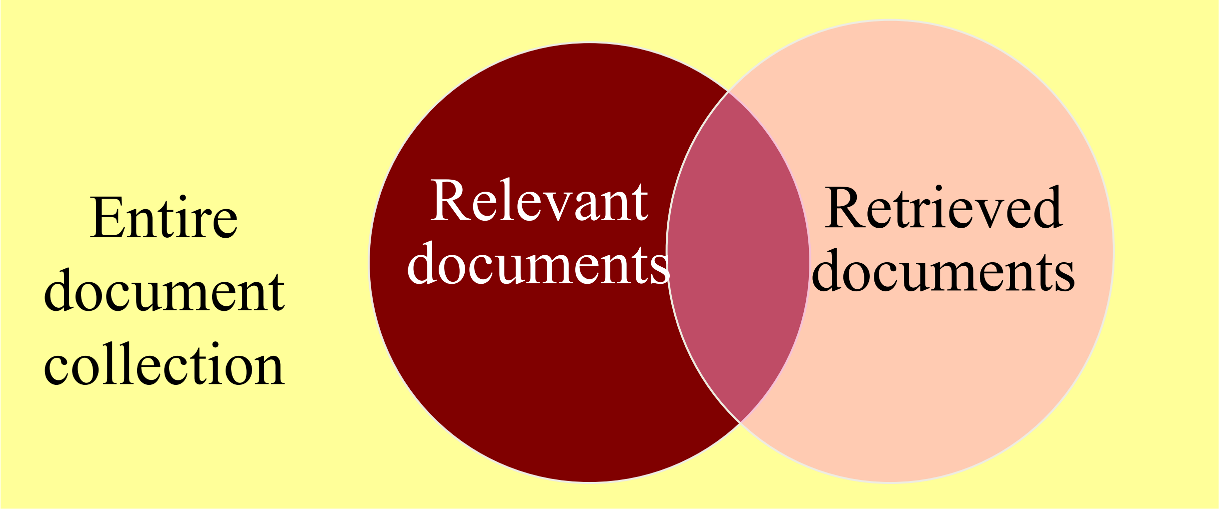
Ranked List Thresholds

- Fixed
  - e.g., keep the first 10 results.

- Score threshold:
  - e.g., keep results with score in the top 5% of all scores.

- Gap threshold:
  - traverse the ranked list (from highest to lowest score)
  - find the widest gap between adjacent scores
  - the score immediately prior to the gap becomes the threshold to cut the list

Precision and Recall



irrelevant	retrieved & irrelevant	not retrieved & irrelevant
relevant	retrieved & relevant	not retrieved but relevant
	retrieved	not retrieved

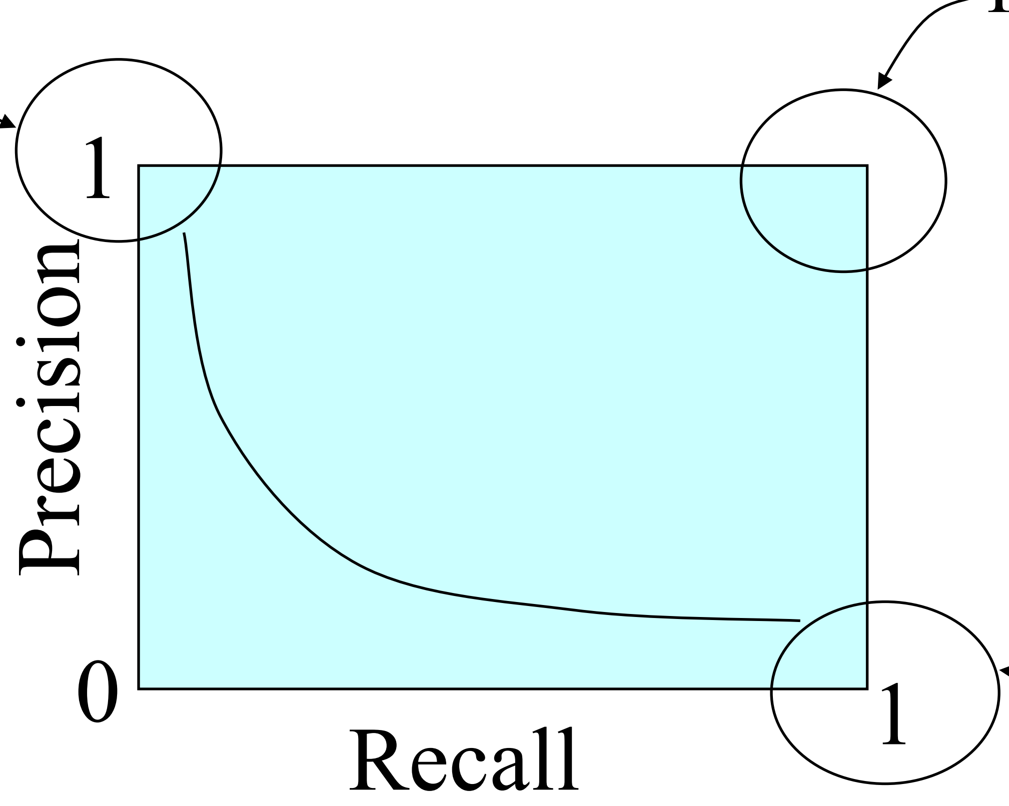
$$recall = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ relevant\ documents}$$

$$precision = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ documents\ retrieved}$$

Trade-off Between Recall and Precision

Returns relevant documents but misses many useful ones too

The ideal



Returns most relevant documents but includes lots of junk

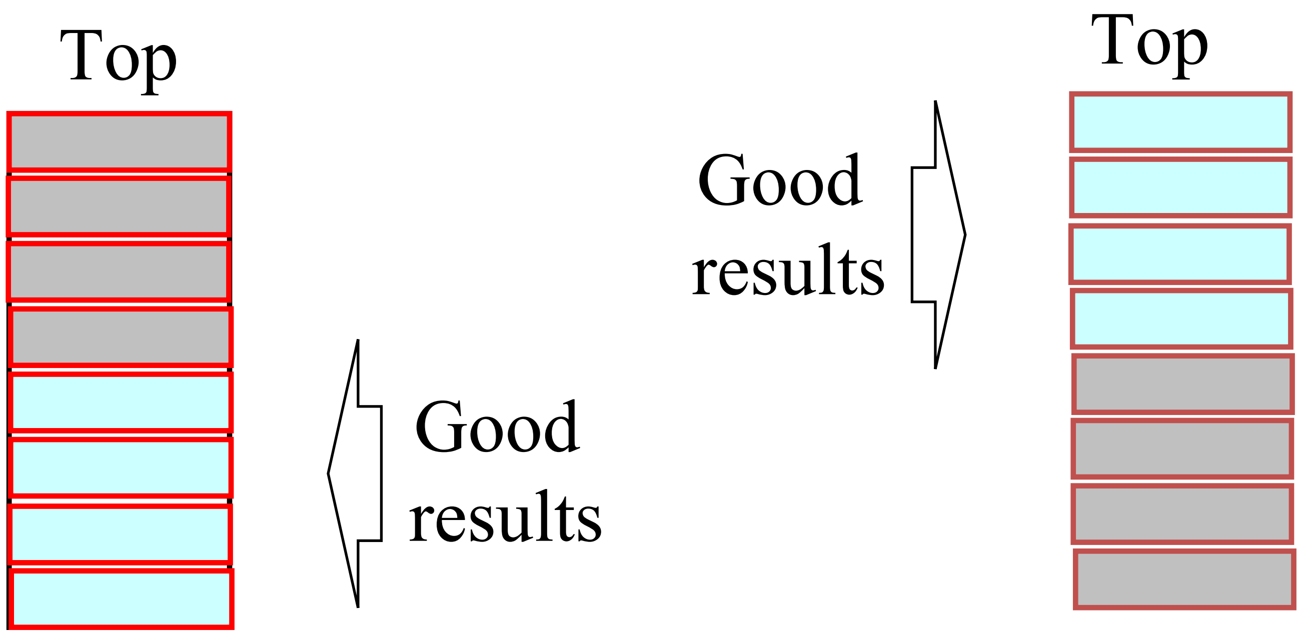
F-Measure

- The traditional F-measure or balanced F-score (**F1 score**) is the harmonic mean of precision and recall

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

Precision and Recall: the Holy Grail

- Precision and recall do not tell the entire story



- Average precision: $\mathrm{AveP} = \frac{\sum_{k=1}^{n}(P(k) \times \mathrm{rel}(k))}{\text{number of relevant documents}}$