# Bug Localization with Semantic and Structural Features using Convolutional Neural Network and Cascade Forest

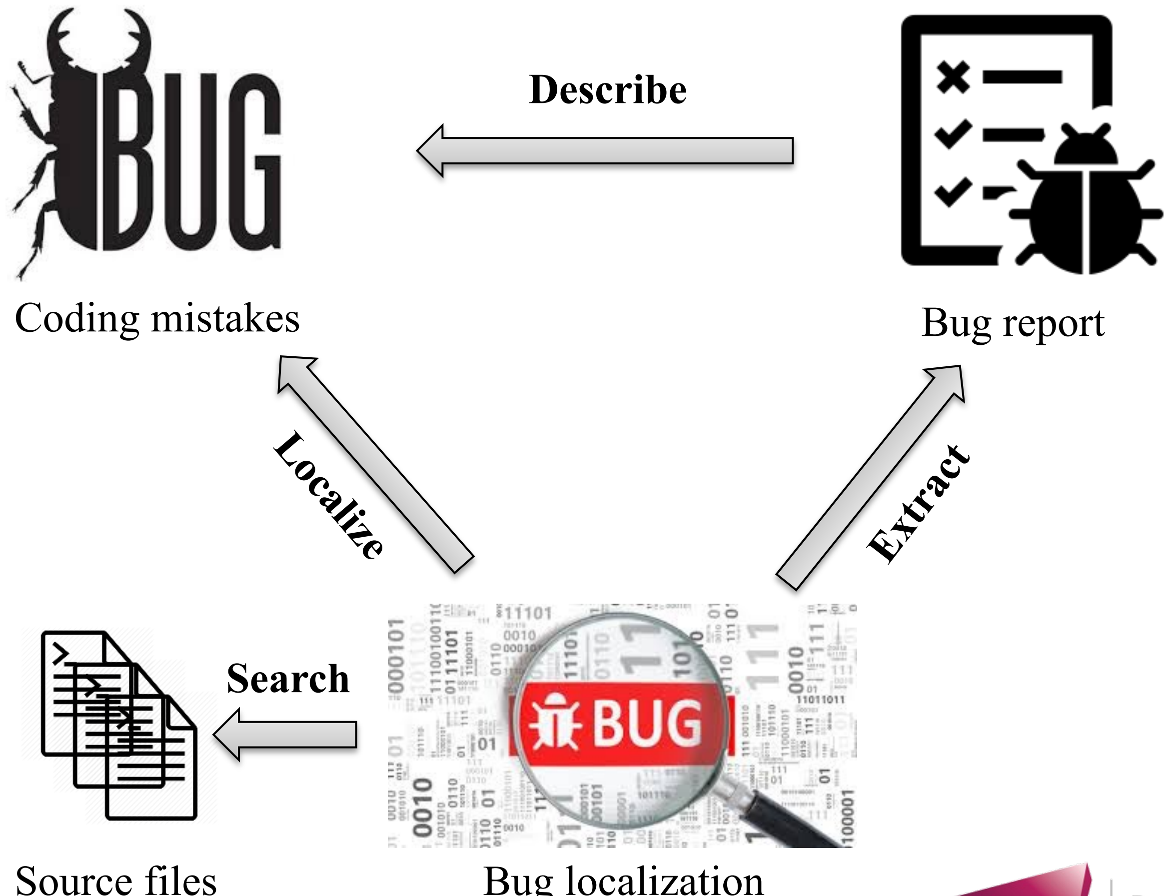
Yan Xiao, Jacky Keung, Qing Mi, Kwabena E. Bennin
Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong

CityU
香港城市大學
City University of Hong Kong
Department of
Computer Science

# Outline

- Background
- Main focus of existing techniques
- CNN_Forest technique
- Experimental results
- Conclusions and Future works

# Background



Describe
Coding mistakes
Bug report
Localize
Extract
Search
Source files
Bug localization

# Outline

- Background
- **Main focus of existing techniques**
- CNN_Forest technique
- Experimental results
- Conclusions and Future works

# Main focus of existing techniques

- Textual similarity

  transform the terms in bug reports and source files into textual representations (vectors) and then measure the textual similarity between them:

  - Latent Semantic Indexing (LSI) represents code and queries as vectors, and estimates the similarity between their vector representations using the cosine similarity.
  - Latent Dirichlet Allocation (LDA)-based approach measures the similarity between the descriptions of bug reports and the topics of source files estimated by LDA.
  - Vector Space Model (VSM) transforms bug reports and source files into feature vectors and then measures the similarity between them.

Textual similarity $\neq$semantics

CityU Department of Computer Science

# Main focus of existing techniques

- The lexical mismatch problem
  - There is a lexical mismatch problem between the texts in bug reports and code tokens in source files caused by the ignorance of the semantic information in them.
  - Word embedding techniques are commonly applied to obtain vectors of words in bug reports and source files, and measures the similarity between them.

**Structural information**

# Main focus of existing techniques

- Lexical mismatch
  - There is lexical mismatch problem between the texts in bug reports and code tokens in source files caused by the ignorance of the semantic information in them.
  - Word embedding techniques are commonly applied to obtain vectors of words in bug reports and source files, and measures the similarity between them.
- Structural features
  - Compared to natural languages, programs contain rich statistical properties and more stringent structural information.
  - NP-CNN employs convolutional neural networks (CNNs) to extract structural features from both bug reports and source files, and uses a fully-connected network to fuse features.


**Semantic + Structural information**

Department of Computer Science

# Outline

- Background
- Main focus of existing techniques
- **CNN_Forest technique**
- Experimental results
- Conclusions and Future works

# CNN_Forest technique

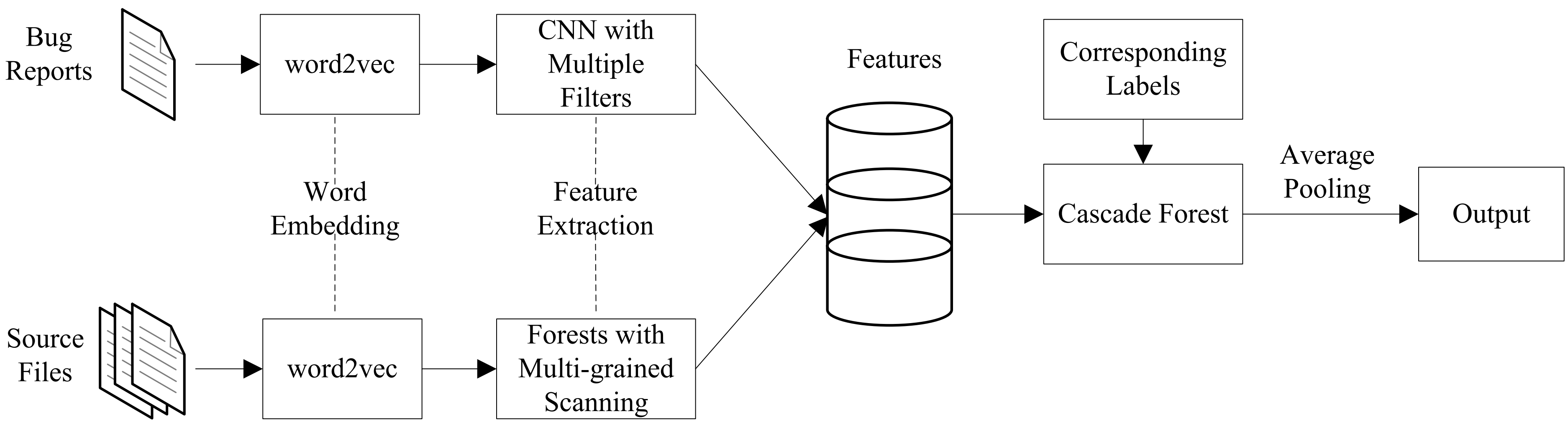



Figure 3: The Overall Workflow of CNN Forest.

# CNN_Forest technique

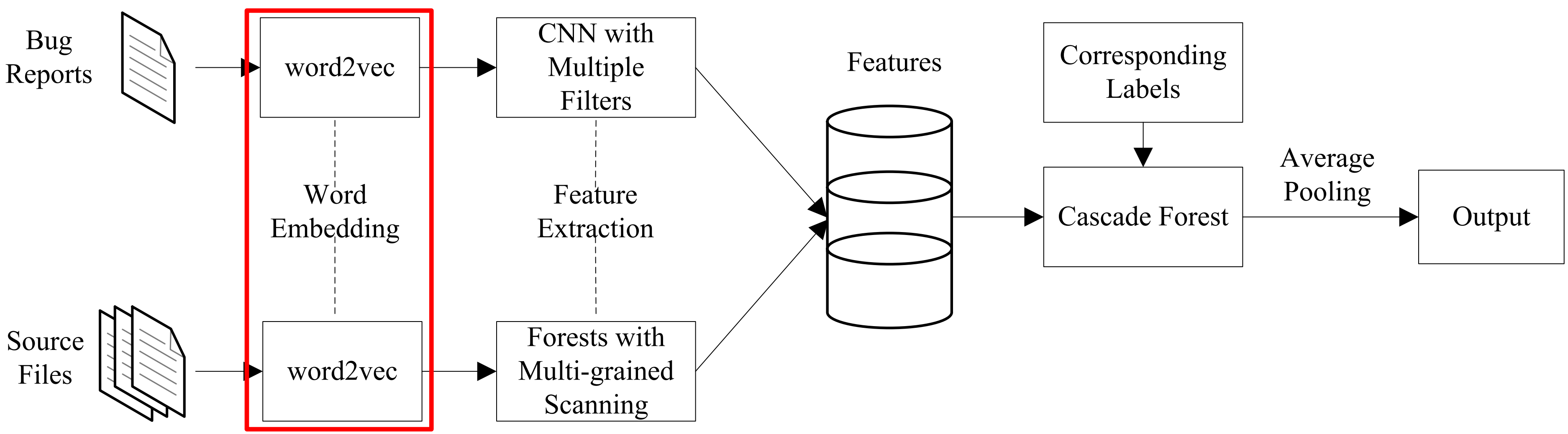



Figure 3: The Overall Workflow of CNN Forest.

# CNN_Forest technique

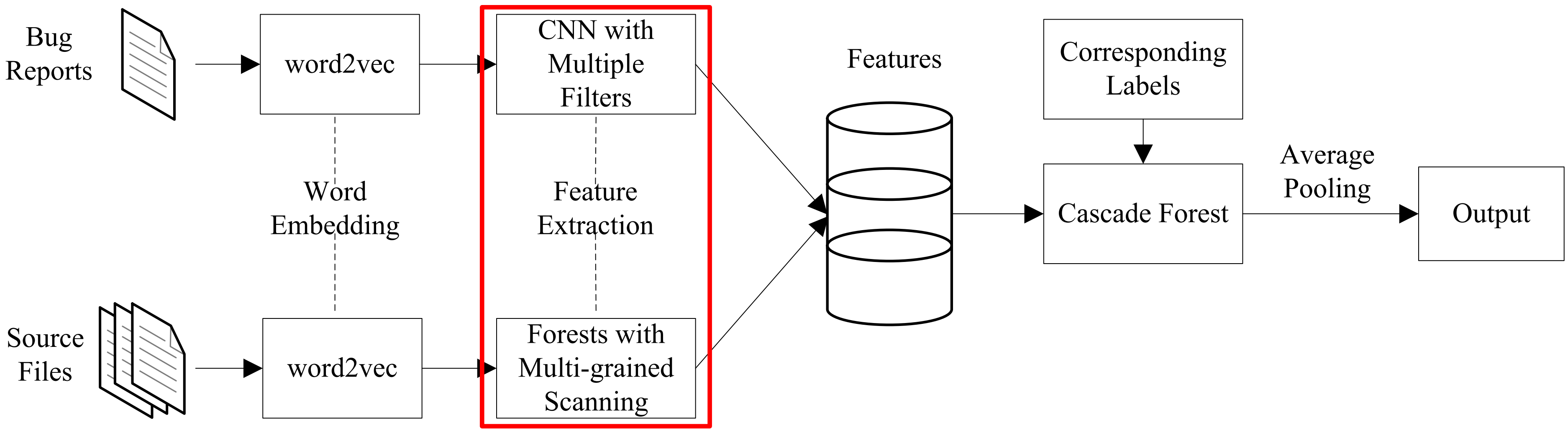



Figure 3: The Overall Workflow of CNN Forest.

# CNN_Forest technique

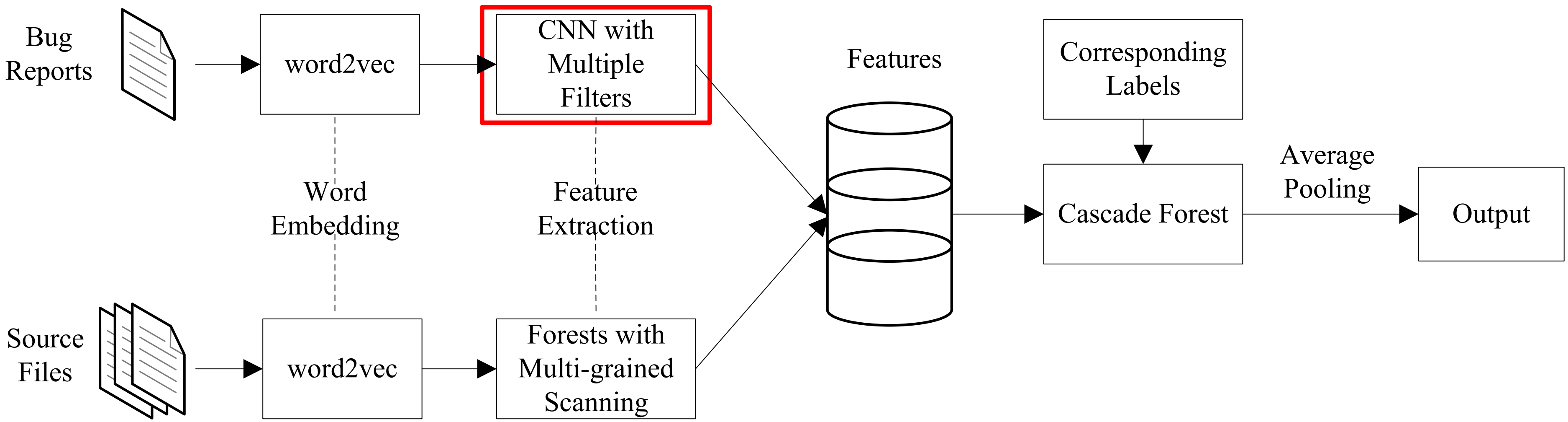



Figure 3: The Overall Workflow of CNN Forest.

# CNN with Multiple Filters to Extract Features from Bug Reports

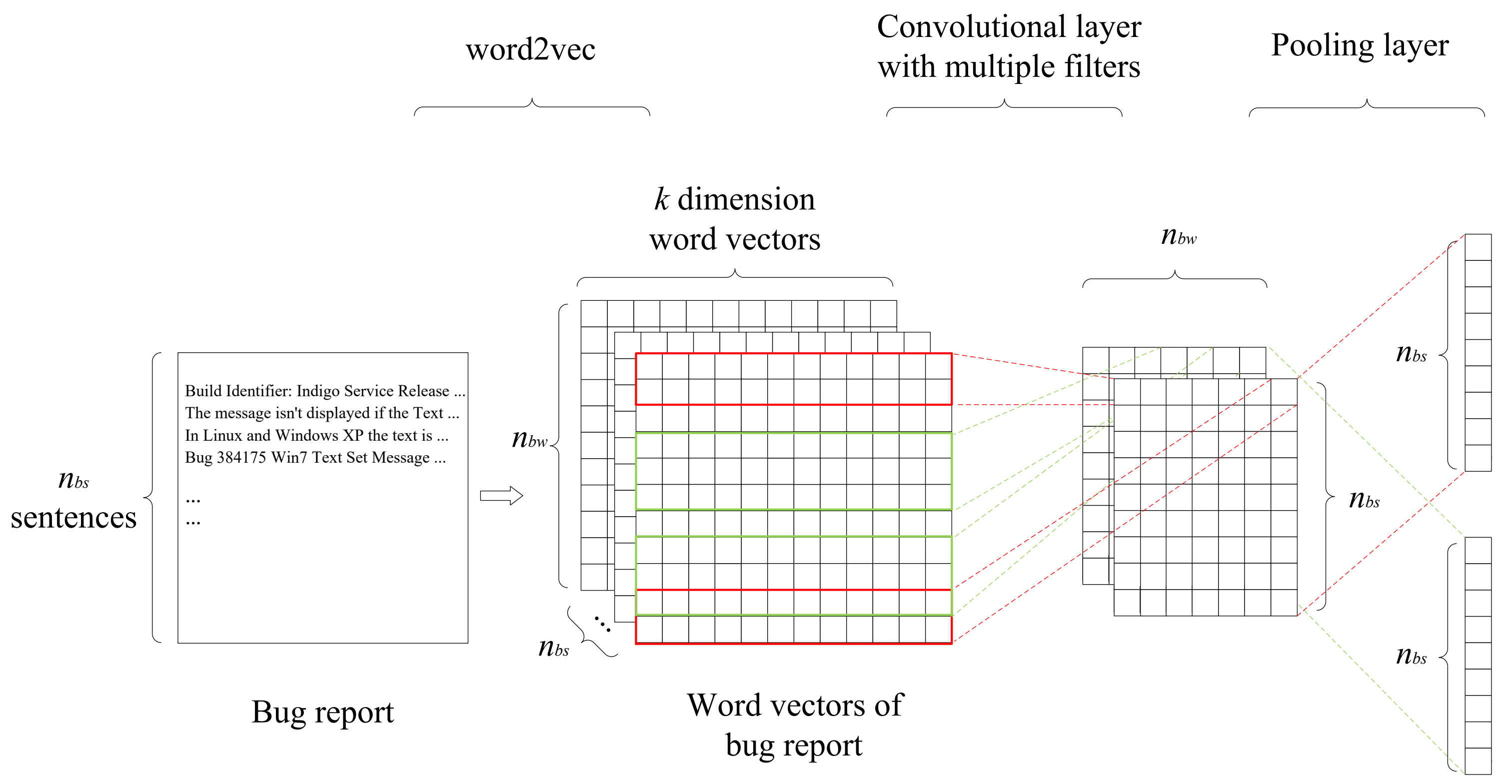



Figure 4: The Feature Extraction of Bug Reports using CNN.

# CNN_Forest technique

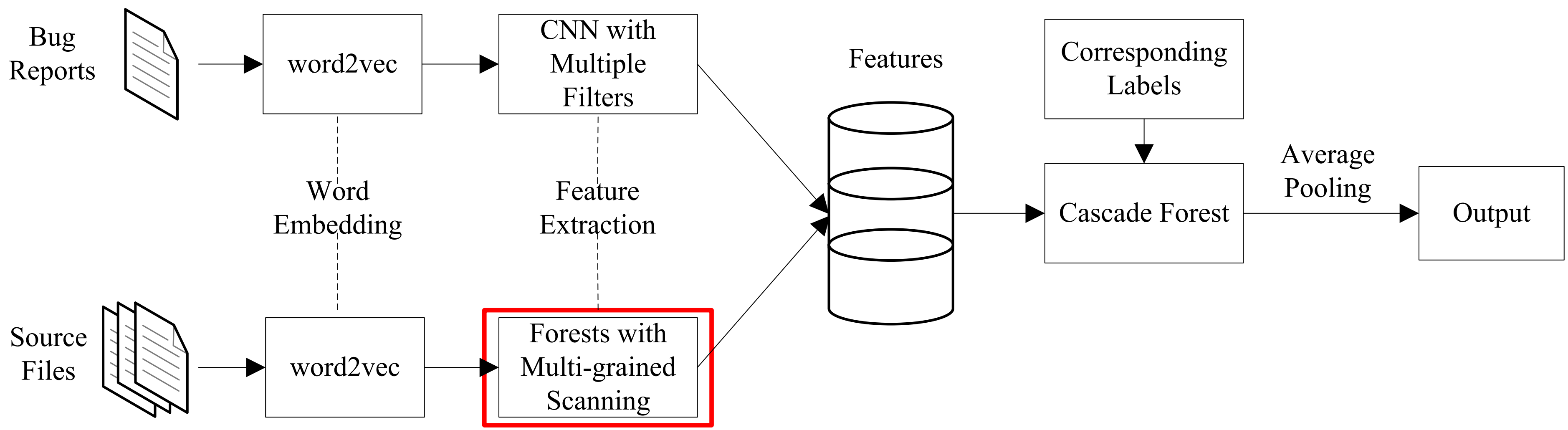



Figure 3: The Overall Workflow of CNN Forest.

# Ensemble of Random Forests with Multi-grained Scanning for Source Files

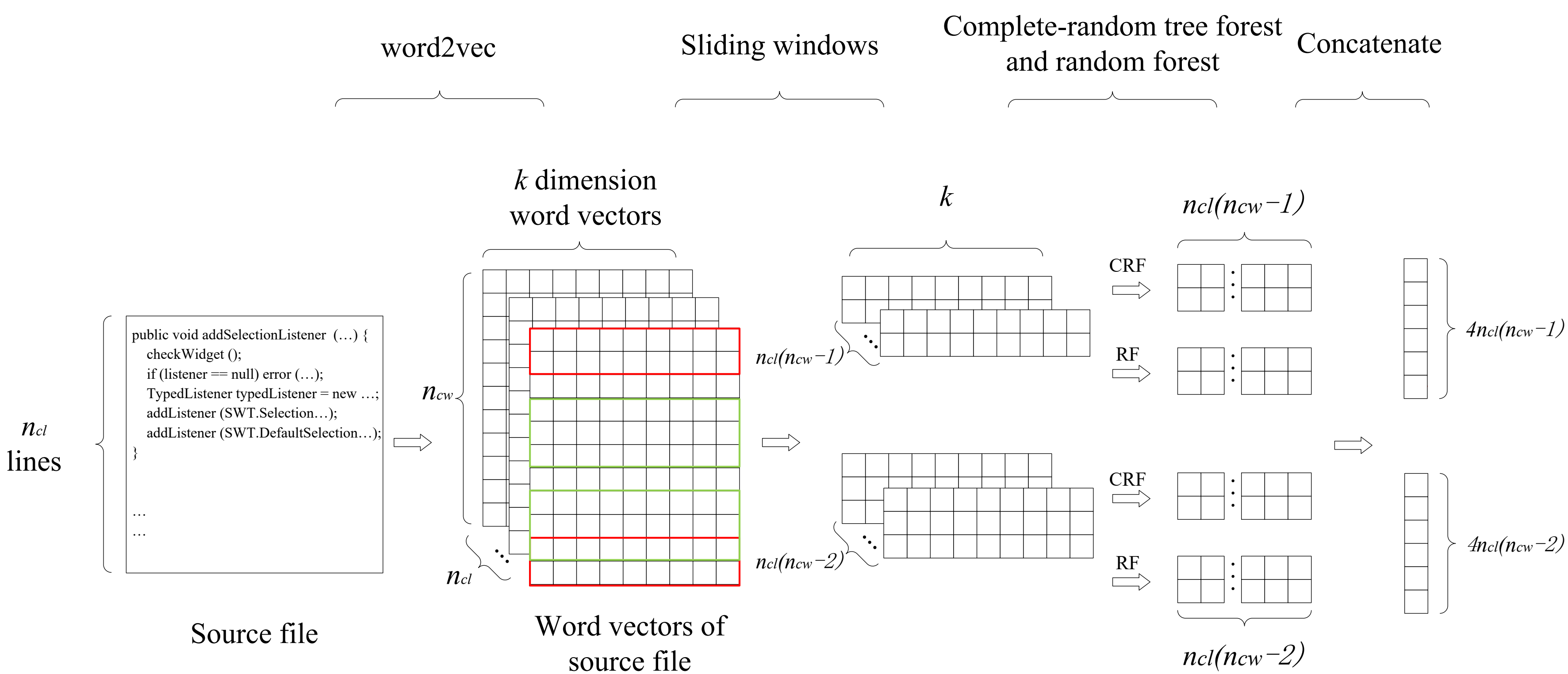



Figure 5: The Feature Extraction of Source Files using Ensemble of Random Forests.

# CNN_Forest technique

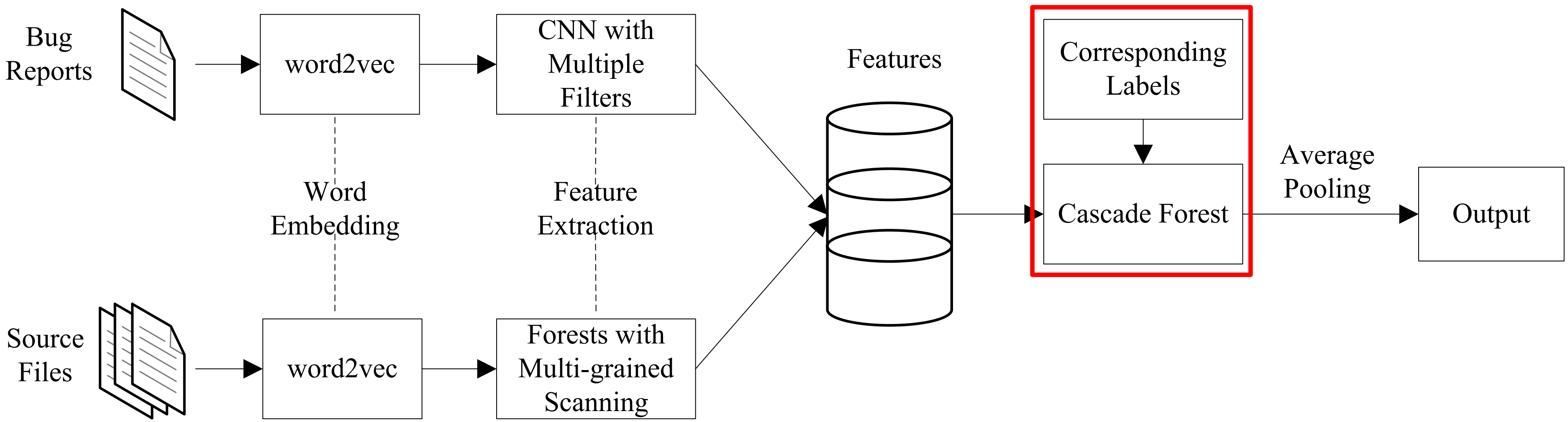



Figure 3: The Overall Workflow of CNN Forest.

# Cascade Forest

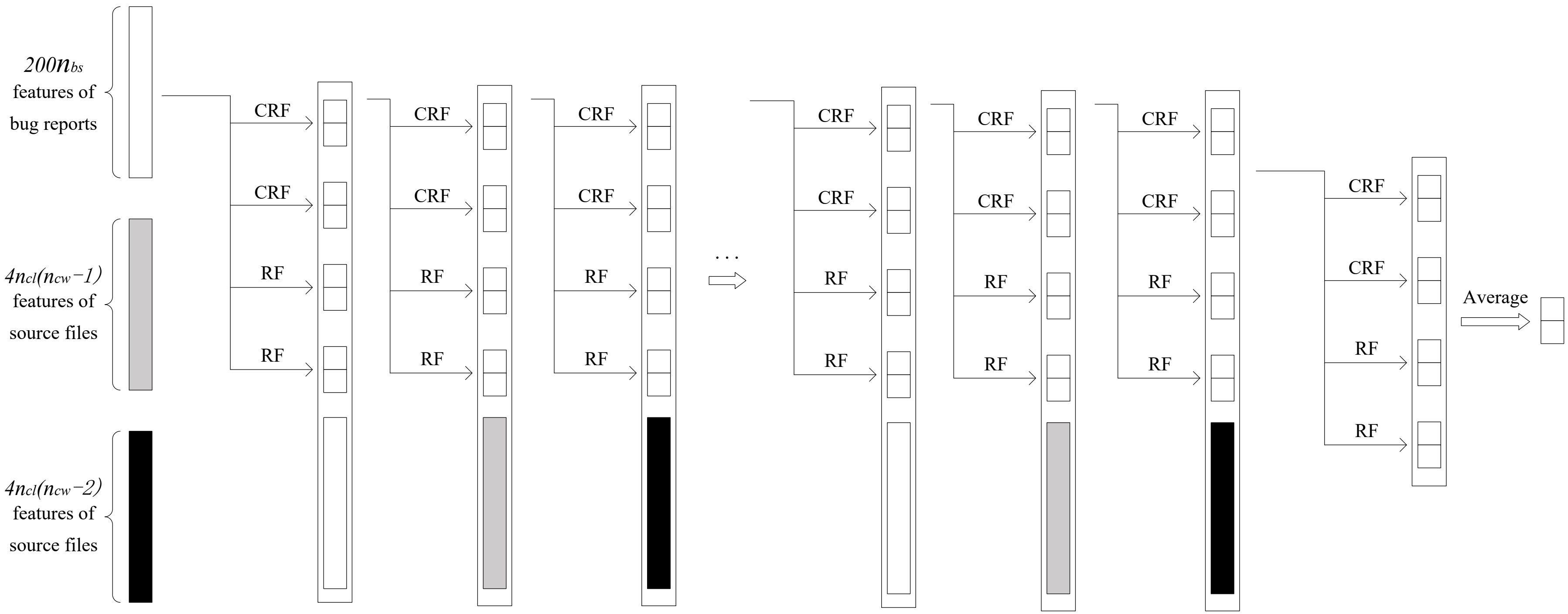



Figure 6: Cascade Forest to Further Extract Features and Learn the Correlated Relationships between Bug Reports and Source Files.

# Outline

- Background
- Main focus of existing techniques
- CNN_Forest technique
- **Experimental results**
- Conclusions and Future works

# Experimental Preparations

Table I Subject Projects

| Project | Time Range | # of Bug Reports for Evaluation | # of Bug Reports for Tuning | # of Bug Reports for Training | # of Bug Reports for Testing |
|---|---|---|---|---|---|
| AspectJ[1] | 03/02-01/14 | 593 | 100 | 400 | 93 |
| Eclipse UI[2] | 10/01-01/14 | 3,656 | 1,500 | 500 | 1,656 |
| JDT[3] | 10/01-01/14 | 2,632 | 1,500 | 500 | 632 |
| SWT[4] | 02/02-01/14 | 2,817 | 1,500 | 500 | 817 |
| Tomcat[5] | 07/02-01/14 | 1,056 | 400 | 500 | 156 |

[1] http://eclipse.org/aspectj/
[2] http://projects.eclipse.org/projects/eclipse.platform.ui
[3] http://www.eclipse.org/jdt/
[4] http://www.eclipse.org/swt/
[5] http://tomcat.apache.org

Metrics

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{first_i}$$

$$MAP = \frac{1}{Q}\sum_{i=1}^{Q}\sum_{r=1}^{R}\frac{(N_T(r)/r)\times ind(r)}{N_B(i)}$$

# Comparisons with other techniques

- validation set:
  - *CNN_CNN* uses the same strategy (CNN) to extract features from bug reports and source files.
  - *Forest_Forest* applies the same ensemble of random forests to extract features of bug reports and source files.
- testing set:
  - *NP-CNN* applies CNN to learn unified features from natural and programming languages.
  - *LR+WE* enhances their previously-proposed learning-to-rank model (LR) by adding new features obtained by word embedding (WE) techniques.
  - *DNNLOC* combines deep learning techniques with the information retrieval techniques to localize the buggy files for bug reports.
  - *BugLocator* measures the textural similarity between the texts in bug reports and source files using the revised Vector Space Model (rVSM).

# Intrinsic Evaluation

Table 2: Performance Comparison for Intrinsic Evaluation.

| Project | Metrics | CNN_Forest | CNN_CNN | Forest_Forest |
|---|---|---|---|---|
| AspectJ | MAP | **0.458** | 0.449 | 0.430 |
| | MRR | **0.563** | 0.560 | 0.549 |
| Eclipse UI | MAP | 0.460 | 0.441 | **0.465** |
| | MRR | **0.590** | 0.561 | 0.588 |
| JDT | MAP | **0.448** | **0.448** | 0.435 |
| | MRR | **0.517** | 0.502 | 0.492 |
| SWT | MAP | **0.462** | 0.415 | 0.439 |
| | MRR | **0.530** | 0.507 | 0.512 |
| Tomcat | MAP | **0.627** | 0.623 | 0.604 |
| | MRR | **0.681** | 0.669 | 0.647 |

# Extrinsic Evaluation

Structural

Semantic Textual

Textual similarity

Table 3: Performance Compa… the State… rt Techn…

| Project | Metrics | CNN_Forest | NP-CNN | LR+WE | DNNLOC | BugLocator |
|---|---|---|---|---|---|---|
| AspectJ | MAP | **0.436** | 0.402 | 0.302 | 0.323 | 0.278 |
| | MRR | **0.519** | 0.487 | 0.454 | 0.475 | 0.364 |
| Eclipse UI | MAP | **0.432** | 0.429 | 0.398 | 0.413 | 0.332 |
| | MRR | **0.534** | 0.529 | 0.461 | 0.514 | 0.383 |
| JDT | MAP | **0.423** | 0.405 | 0.417 | 0.342 | 0.290 |
| | MRR | 0.514 | 0.463 | **0.516** | 0.452 | 0.367 |
| SWT | MAP | **0.394** | 0.371 | 0.381 | 0.369 | 0.269 |
| | MRR | **0.482** | 0.466 | 0.446 | 0.445 | 0.312 |
| Tomcat | MAP | **0.550** | 0.529 | 0.503 | 0.523 | 0.425 |
| | MRR | **0.614** | 0.585 | 0.556 | 0.604 | 0.481 |

# Why does CNN_Forest work best?

- Word embedding
  - Convert the texts in bug reports and source files into word vectors.
- CNN with multiple filters
  - CNN has performed excellently in natural language processing because of convolving filters.
- Ensemble of random forests with multi-grained scanning
  - Source files are composed of code tokens that involve more stringent structural information.
- Cascade forest
  - Correlated relationships between bug reports and source files are learned by the alternate cascade structure that is similar to the layer-structure in deep learning.

# Outline

- Background
- Main focus of existing techniques
- CNN_Forest technique
- Experimental results
- **Conclusions and Future works**

CityU Department of Computer Science

# Conclusions

- This paper proposed CNN_Forest, a new CNN and random forest-based model for bug localization.
- Two **different techniques** are leveraged to extract features respectively from bug reports and source code files **automatically** according to the **differences** between natural languages and programming languages.
- Both **semantic and structural information** of bug reports and source files are extracted. The ensemble of random forests is applied to detect the structural information from the source code. The alternate cascade forest works as the **layer-structure** in deep learning to learn the correlated relationships between bug reports and source files.

# Future works

- Fine-tune the proposed model to further improve its process automation and prediction performance for bug localization.
  - evaluation on other kinds of random forest techniques (e.g., extra-trees) to enhance diversity.
- Focus on projects written in other programming languages.
- Extend the model to aforementined projects for bug localization with other performance metrics to evaluate their performances.
  - Accuracy@k
  - Area Under the receiver operator characteristic Curve (AUC)
- Examine the performance of the proposed model in other applications of software engineering
  - defect prediction

# Thank you for your attention!

香港城市大學
City University of Hong Kong

Department of
Computer Science