A Large-Scale Comparative Evaluation of IR-based Tools for Bug Localization

SHAYAN AKBAR

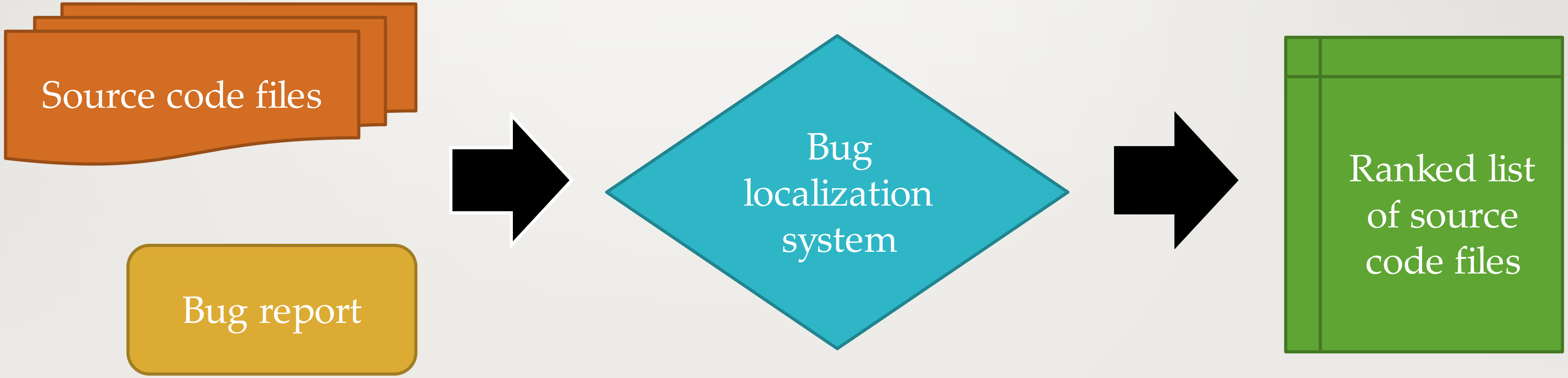
AVI KAK

PURDUE UNIVERSITY

[AN MSR 2020 PRESENTATION]

We focus on the problem of IR-based bug localization

- Given a large corpus of source code files belonging to a software repository and a bug report, find a list of source code files ranked based on their relevancy to the bug report



Contents

- Motivation / Introduction
- The three generations of IR-based bug localization
- Eight retrieval algorithms selected from the three generations to study
- Bugzbook --- A large, diverse bug localization dataset
- Results
- Conclusion

Why yet another study on bug localization?

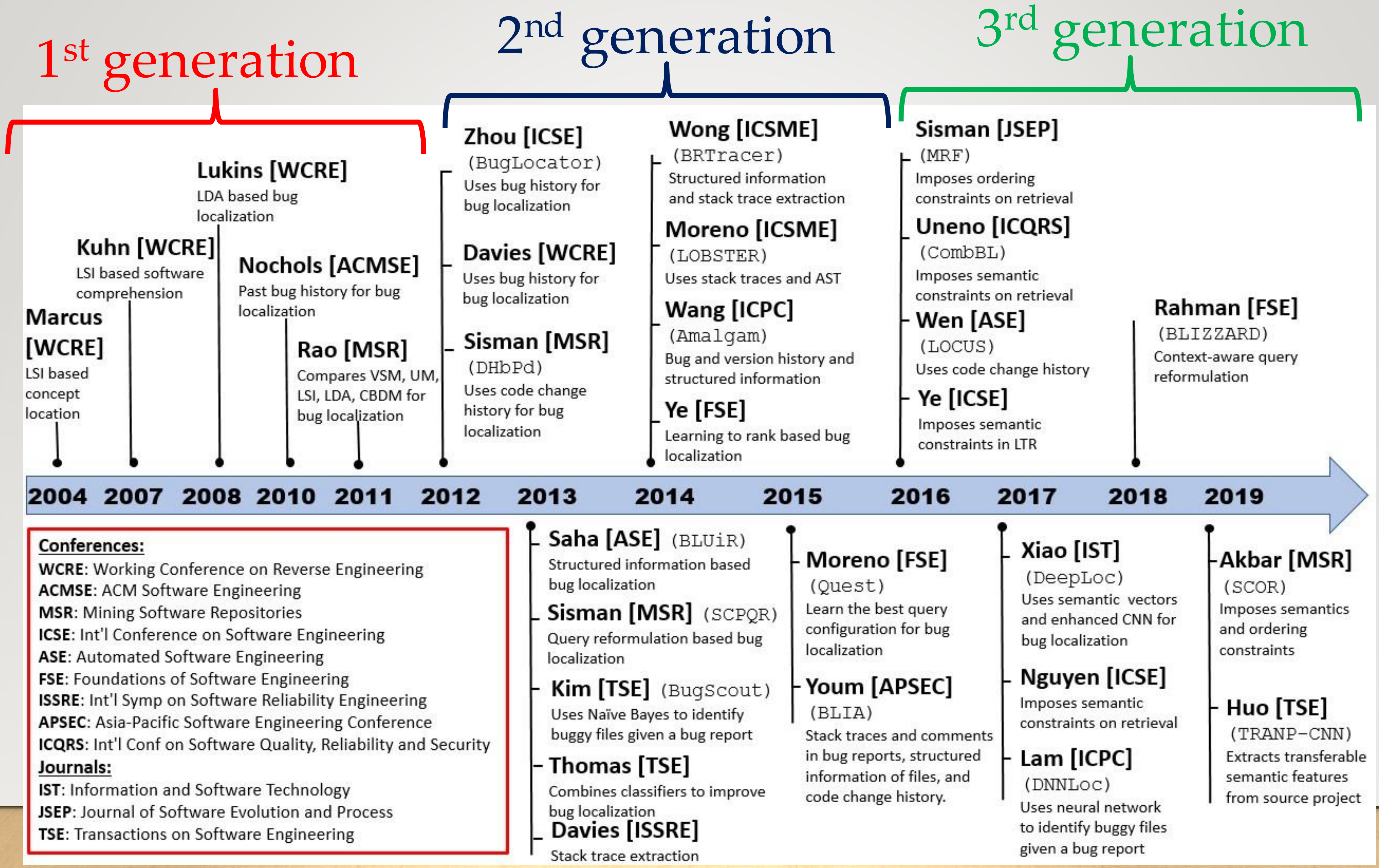
- **Because previous studies …**
  - mostly relied only on Java software repositories
  - are based on datasets with only few thousand bug reports
  - conducted experiments only on very few software projects
- **We created Bugzbook --- a new large and diverse bug localization dataset containing over 20000 bug reports**
  - **These bug reports belong to 29 different software projects**
  - **The projects belong to Java, C/C++ and Python programming languages**

We experimented with past bug localization algorithms on Bugzbook

- We divided the past 15 years of studies in IR-based bug localization into three generations
- Each generation tells a story of what happened in the field of bug localization during that generation
- We compiled a list of the prominent papers published in each generation
- From all the three generations we selected eight algorithms and conducted experiments on Bugzbook dataset to evaluate their relative performances

The three generations of IR-based bug localization

(spanning studies from last 15 years)


1st generation
2nd generation
3rd generation
Marcus [WCRE]
LSI based concept location
Kuhn [WCRE]
LSI based software comprehension
Lukins [WCRE]
LDA based bug localization
Nochols [ACMSE]
Past bug history for bug localization
Rao [MSR]
Compares VSM, UM, LSI, LDA, CBDM for bug localization
Zhou [ICSE]
(BugLocator)
Uses bug history for bug localization
Davies [WCRE]
Uses bug history for bug localization
Sisman [MSR]
(DHbPd)
Uses code change history for bug localization
Wong [ICSME]
(BRTracer)
Structured information and stack trace extraction
Moreno [ICSME]
(LOBSTER)
Uses stack traces and AST
Wang [ICPC]
(Amalgam)
Bug and version history and structured information
Ye [FSE]
Learning to rank based bug localization
Sisman [JSEP]
(MRF)
Imposes ordering constraints on retrieval
Uneno [ICQRS]
(CombBL)
Imposes semantic constraints on retrieval
Wen [ASE]
(LOCUS)
Uses code change history
Ye [ICSE]
Imposes semantic constraints in LTR
Rahman [FSE]
(BLIZZARD)
Context-aware query reformulation
2004
2007
2008
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Saha [ASE] (BLUiR)
Structured information based bug localization
Sisman [MSR] (SCPQR)
Query reformulation based bug localization
Kim [TSE] (BugScout)
Uses Naïve Bayes to identify buggy files given a bug report
Thomas [TSE]
Combines classifiers to improve bug localization
Davies [ISSRE]
Stack trace extraction
Moreno [FSE]
(Quest)
Learn the best query configuration for bug localization
Youm [APSEC]
(BLIA)
Stack traces and comments in bug reports, structured information of files, and code change history.
Xiao [IST]
(DeepLoc)
Uses semantic vectors and enhanced CNN for bug localization
Nguyen [ICSE]
Imposes semantic constraints on retrieval
Lam [ICPC]
(DNNLoc)
Uses neural network to identify buggy files given a bug report
Akbar [MSR]
(SCOR)
Imposes semantics and ordering constraints
Huo [TSE]
(TRANP-CNN)
Extracts transferable semantic features from source project
Conferences:
WCRE: Working Conference on Reverse Engineering
ACMSE: ACM Software Engineering
MSR: Mining Software Repositories
ICSE: Int'l Conference on Software Engineering
ASE: Automated Software Engineering
FSE: Foundations of Software Engineering
ISSRE: Int'l Symp on Software Reliability Engineering
APSEC: Asia-Pacific Software Engineering Conference
ICQRS: Int'l Conf on Software Quality, Reliability and Security
Journals:
IST: Information and Software Technology
JSEP: Journal of Software Evolution and Process
TSE: Transactions on Software Engineering

1st generation [2004 – 2011]

- **Most of the early works in bug localization relied on using traditional BoW (Bag of Words) based IR tools to rank files based on their relevancy to the bug report**
- Tradition BoW IR techniques:
  - UM (Unigram Model)
  - TFIDF (Term Frequency Inverse Document Frequency)
  - DLM (Dirichlet Language Model)
  - VSM (Vector Space Model)
  - LSI (Latent Semantic Indexing)
  - …

2nd generation [2011 – 2016]

- **Software-centric information modelled in bug localization systems**
- Information derived from:
  - bug report history
  - version history
  - structure of bug reports
  - structure of code files
  - …
- BugLocator, BLUiR, DHbPd, BRTracer, BLIA, LOBSTER, and Amalgam are some of the tools developed during this generation

3rd generation [2016 – present]

- **Term-term proximity, order and semantic relationships modelled in bug localization systems**
- **<u>Proximity</u>:** Code files that contain bug report terms in similar proximities as in the bug report itself are more likely to be relevant to the bug report
- **<u>Order</u>:** Code files should be ranked higher in the list if they contain bug report terms in the same order as they appear in the bug report text
- **<u>Semantics</u>:** In addition to exact matching between terms in the code files and bug report, terms should also be matched based on semantics

From the three generations we selected eight retrieval algorithms to study

1. **TFIDF** (Term Frequency Inverse Document Frequency)
2. **DLM** (Dirichlet Language Model)

1st generation

3. **BugLocator**
4. **BLUiR** (Bug Localization Using Information Retrieval)

2nd generation

5. **MRF-SD** (Markov Random Fields – Sequential Dependence)
6. **MRF-FD** (Markov Random Fields –Full Dependence)
7. **PWSM** (Per Word Semantic Model)
8. **SCOR** (Source Code Retrieval with Semantics and Order)

3rd generation

1^{st} Generation Tools (Simple BoW Models)

- **TFIDF**
  - works by combining frequencies of query terms in the file (TF) and the inverse document frequencies of query terms in the corpus (IDF) to determine the relevance of a file to a query

$$score_{tfidf}(f, Q) \propto tf(q_i, f) * idf(q_i)$$

- **DLM**
  - a probabilistic model that estimates a smoothed first order probability distribution of the query terms in the file to produce the relevance score for a file to a query

$$score_{dlm}(f, Q) \propto tf(q_i, f) * cf(q_i)$$

2nd Generation Tools (Software-centric information)

- **BugLocator**
  - takes into account the history of the past bug reports and leverages bug reports that have been previously fixed to improve bug localization
- **BLUiR**
  - extracts code entities such as classes, methods, variables, and comments from code files to help in localizing a buggy file

3rd Generation Tools (Modelling Proximity and Order)

- **MRF-SD**
  - measures the probability distribution of the frequencies of pairs of **consecutively** occurring query terms appearing in the file to compute the relevance score for the file to a query

$$score_{sd}(f, Q) \propto tf(q_i q_{i+1}, f) * cf(q_i q_{i+1})$$

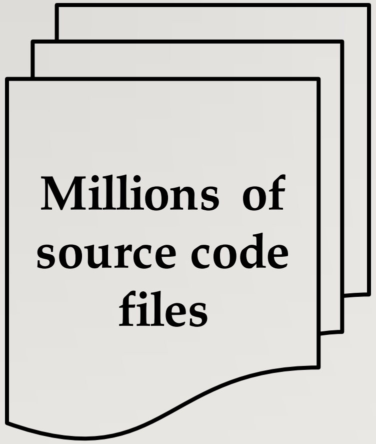
- **MRF-FD**
  - is a term-term dependency model that considers frequencies of **all** pairs of query terms appearing in the file to determine the relevance of the file to the query

$$score_{fd}(f, Q) \propto tf(q_i q_j, f) * cf(q_i q_j)$$

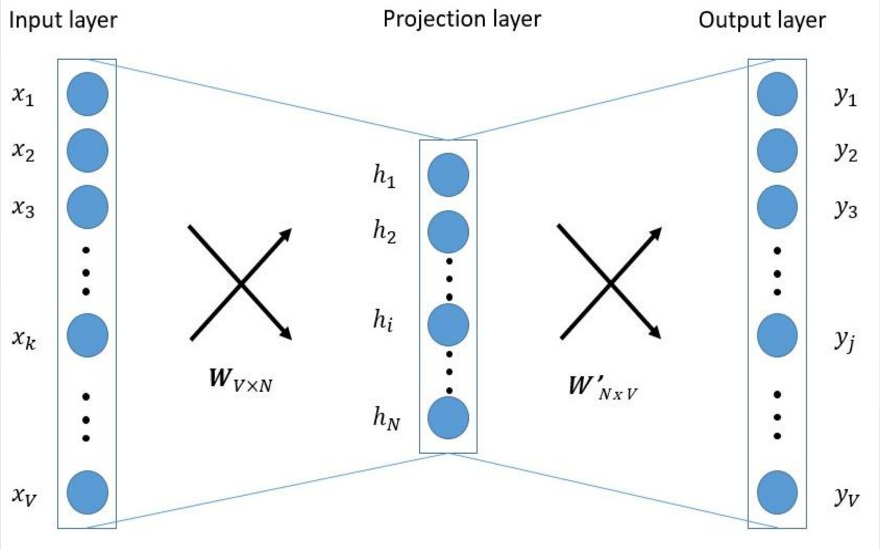
3rd Generation Tools (Modeling semantics)

- In order to discuss how to model semantics in retrieval process first we need to discuss semantic word embeddings

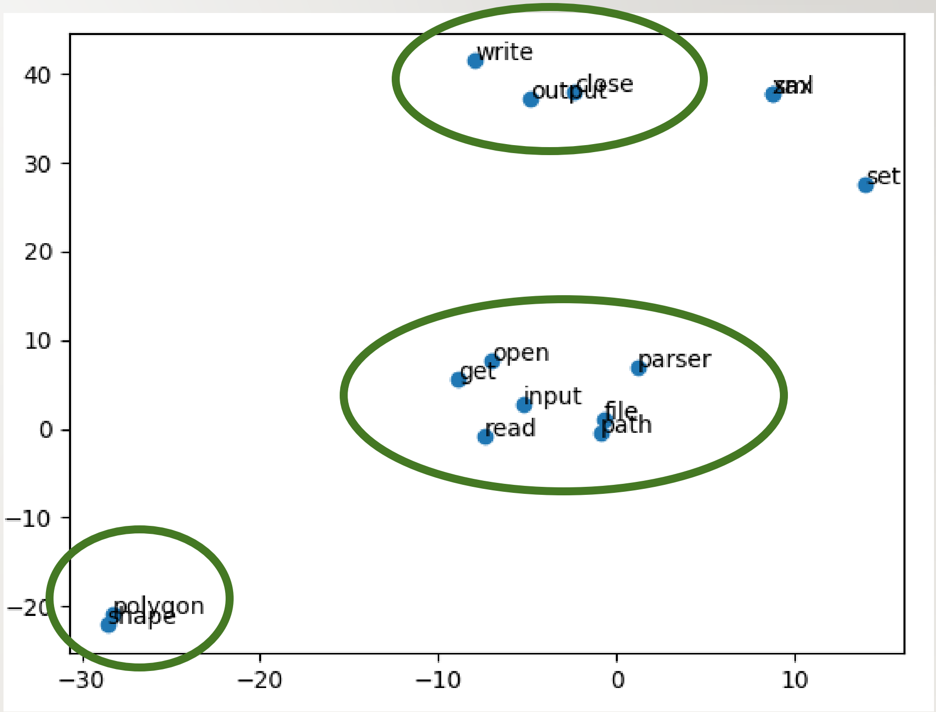
3rd Generation Tools
(word2vec for constructing semantic word embeddings)



word2vec neural network



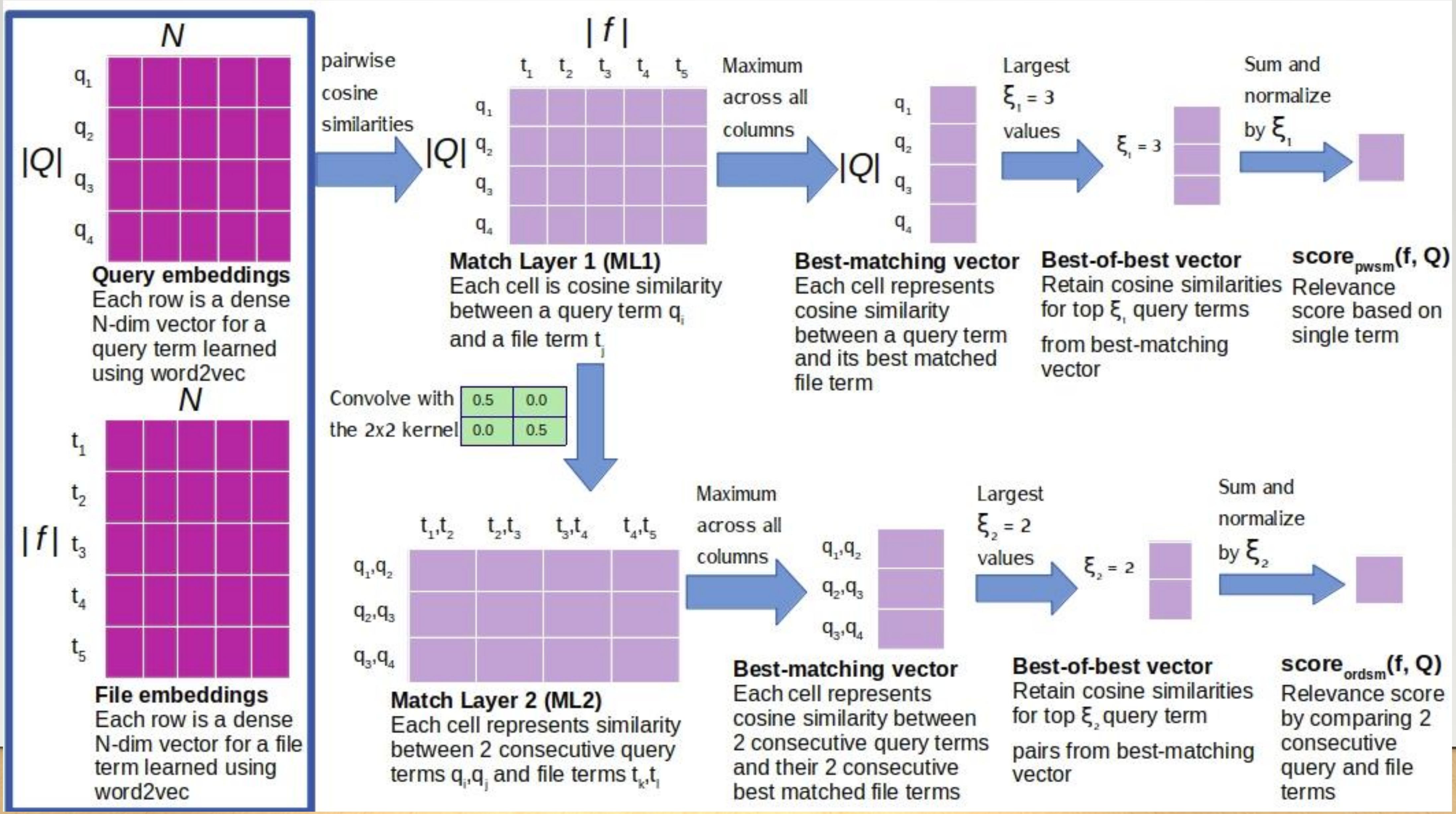
Semantic word vectors



SCOR Word Embeddings

https://engineering.purdue.edu/RVL/SCOR_WordEmbeddings/

3rd Generation Tools (Modelling Semantics as in SCOR)



3rd Generation Tools (Modelling Semantics)

- **PWSM:**
  - uses embeddings derived from word2vec algorithm to model term-term contextual semantic relationships in retrieval process

$$score_{pwsm}(f,Q) = \alpha \,.\, score_{dlm}(f,Q) + \beta \,.\, score_{pwsm}(f,Q)$$

- **SCOR:**
  - combines MRF based term-term dependency modeling with semantic word embeddings as made possible by word2vec algorithm to improve bug localization

$$score_{scor}(f,Q) = \alpha \,.\, score_{dlm}(f,Q) + \beta \,.\, score_{sd}(f,Q) + \mu \,.\, score_{pwsm}(f,Q) + \Omega \,.\, score_{ordsm}(f,Q)$$

Components built into each of the eight algorithms

	TFIDF	DLM	BugLocator	BLUiR	MRF SD	MRF FD	PWSM	SCOR
BoW	✓	✓	✓	✓	✓	✓	✓	✓
Order					✓	✓		✓
Semantics							✓	✓
Stack Trace					✓	✓	✓	✓
Structure				✓				
Bug History			✓					

We experimented with these eight algorithms on Bugzbook (features of Bugzbook dataset)

- **Bugzbook contains over 20000 bug reports from 29 projects belonging to Java, C/C++, and Python projects**
- **We processed over 4 million source code files belonging to all the 29 projects**
- Out of 29 projects, 23 projects belong to Apache community (Ambari, Spark, Camel, …)
- We downloaded Apache bug reports from well-managed Jira platform
- Also present in Bugzbook are bug reports from other large-scale projects (OpenCV, Pandas, …)
- The bug reports for non-Apache projects were downloaded from GitHub

How Bugzbook compare with other datasets?

Dataset	Number of projects	Number of bug reports
moreBugs	2	~400
BUGLinks	2	~4000
iBUGS	3	~400
Bench4BL	46	~10000
Bugzbook	29	~20000

Bugzbook stats

(number of bug reports)

Apache Java projects

Project	#bugs
Ambari	2253
Bigtop	5
Camel	2308
Cassandra	514
CXF	1795
Drill	800
Hbase	2476
Hive	2221

Project	#bugs
JCR	457
Karaf	390
Mahout	162
Math	17
OpenNLP	84
PDFBox	1163
PIG	47
SOLR	471

Project	#bugs
Spark	185
Sqoop	201
Tez	177
Tika	183
Wicket	567
WW	87
Zookeeper	20

Other Java projects

Project	#bugs
Eclipse	4035
AspectJ	291

C/C++, Python projects

Project	#bugs
Chrome	147
OpenCV	8
Pandas	179
Tensorflow	10

Total

#projects	29
#bugs	21253

Eclipse and Chrome bug reports taken from BUGLinks dataset
AspectJ bug reports taken from iBUGS dataset

Bugzbook construction

1. **Gather raw bug reports and code files**

   Download bug reports from Jira and GitHub, and source code files from project archives

2. **Filter raw bug reports (remove duplicates, etc.)**

   Remove bug reports with duplicate tags. Also, remove reports that are not tagged as “bug”.

3. **Link bug reports with respective code files**

   Use commits to identify which bugs it fixes. Also, use modified associated with the commits as relevant files for the respective bug.

4. **Match bug reports with respective versions**

   Bug reports come with a tag that tells which version was affected by the bug

5. **Manual verification of dataset**

   Verify randomly chosen bugs for each project. Check if bug ID, title, description, fixed files are correctly recorded in Bugzbook.

Results on Bugzbook dataset

Highlights of the results

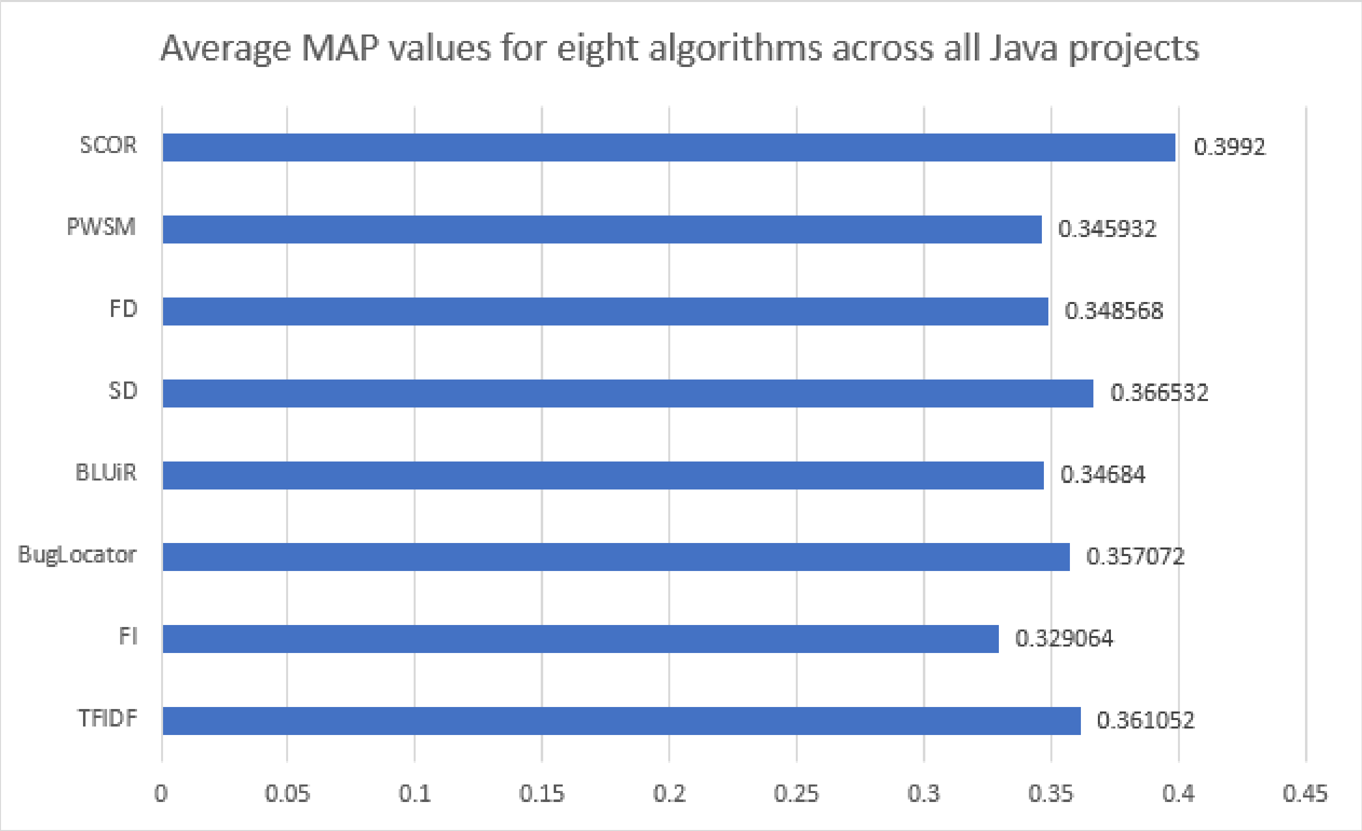
- Experiment 1 (Performance of retrieval algorithms belonging to different generations)
  - Results on Java only projects
  - Results on C/C++ and Python projects
  - Results on all projects
- Experiment 2 (Effect of semantic word embeddings)
  - Can we cross-utilize word embeddings across different programming languages?
  - Does changing the size of word vectors and the word embedding algorithm affect retrieval?

Highlights of the results (contd.)

- **3rd generation tools are the most effective in terms of retrieval precision**
- **SCOR word embeddings (trained on Java language) can be used for searching in C/C++ and Python projects**
- **The size (500, 1000, 1500 dimensions) and type (word2vec, GloVe, FastText) of word embeddings do not affect the performance of retrieval**

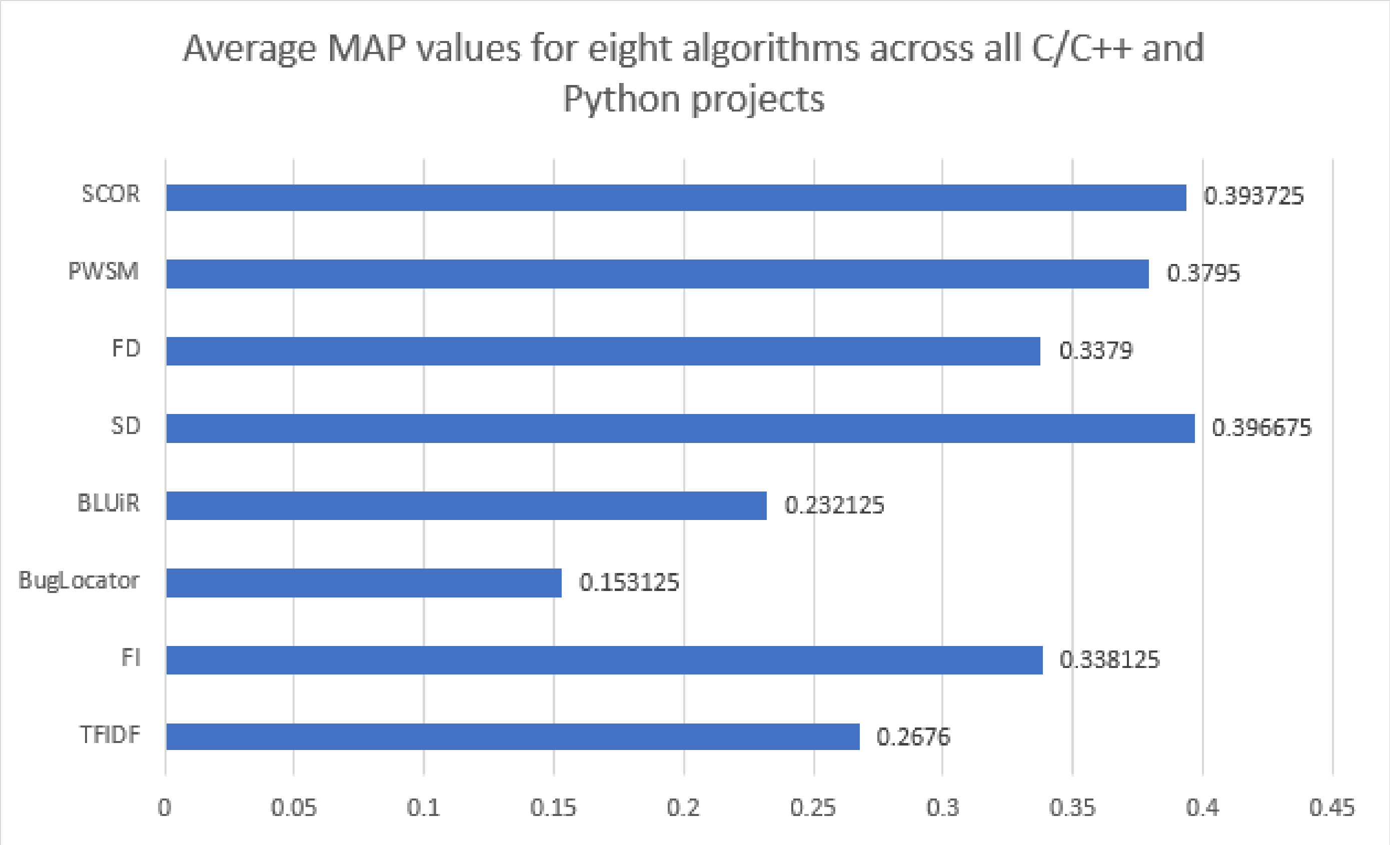
Results for Java projects

SCOR, a third generation tool, outperforms all other algorithms



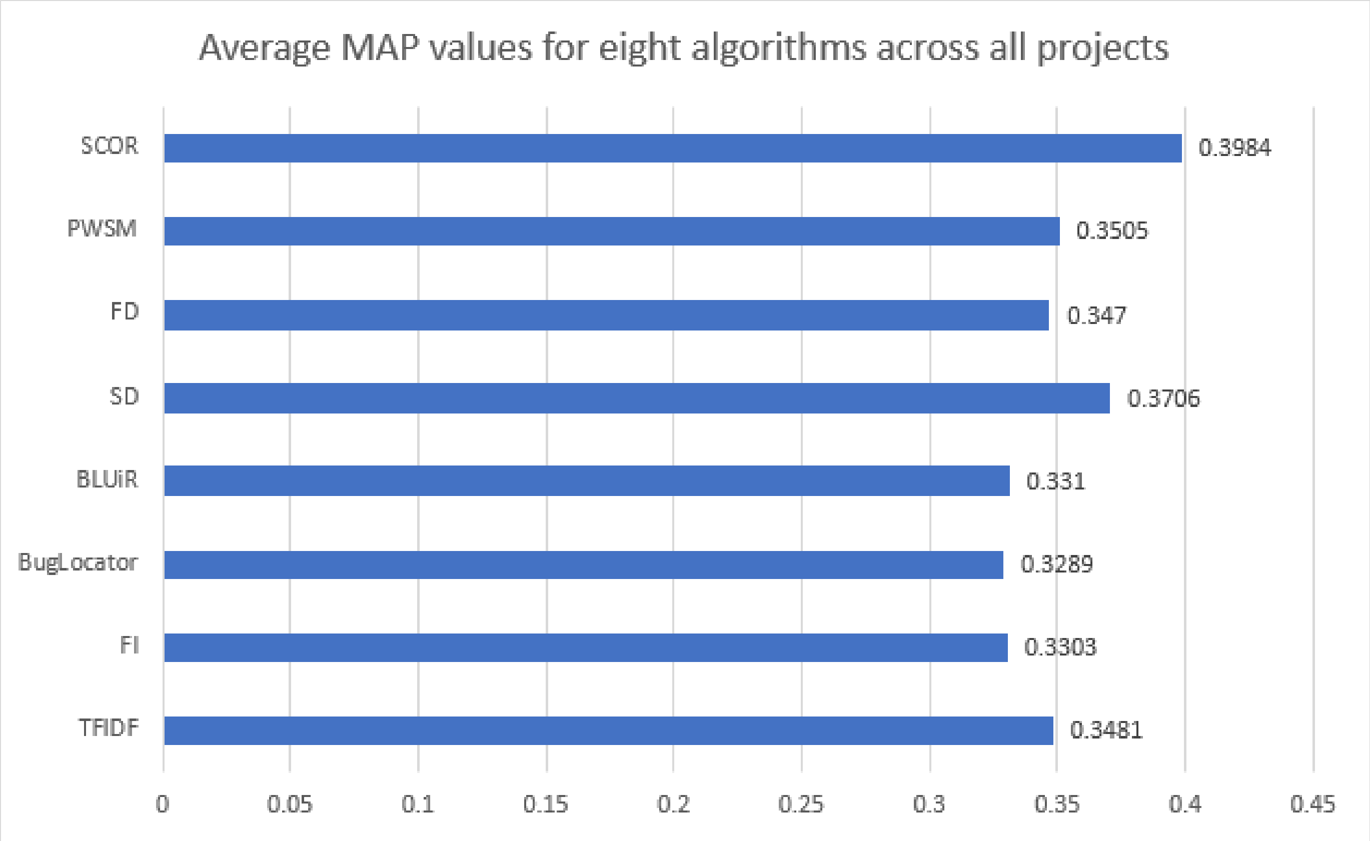
Results for C/C++ and Python projects

SCOR and MRF SD, both third generation tools, outperforms all other algorithms



Results for all projects

SCOR, a third generation tool, outperforms all other algorithms



Cross-utilization of word embeddings

- SCOR word embeddings were trained on Java-only projects
- **The following 5 Java projects present in Bugzbook dataset were not present in the SCOR dataset:**
  - AspectJ, Bigtop, OpenNLP, PDFBox, and Drill
- Therefore, the word embedding algorithm was not trained on these 5 projects
- **However, the bug localization precision on these 5 datasets increases when semantic word embeddings based retrieval algorithms --- PWSM and SCOR --- are used for retrieval.**
- Therefore, the SCOR word embeddings are generic enough to be applied for retrieval in new projects
- **We also observe the same results on the following C/C++ and Python projects**
  - Chrome, OpenCV, Pandas, and Tensorflow

Effect of changing word vector size and word embedding algorithm

- **We observe that the word vector size does not affect the retrieval precision of SCOR when tested on 4000 bug reports of Eclipse dataset**

SCOR500	SCOR1000	SCOR1500
0.3191	0.3204	0.3193

SCOR500 means word vectors having 500 dimensions used for retrieval

- We also trained GloVe and FastText (the two semantic word embedding algorithms that compete with word2vec) on the same SCOR dataset, and used the resulting word vectors for retrieval with SCOR
- **We notice that the change in word embedding algorithm does not affect the retrieval performance of SCOR**

SCOR (word2vec)	SCOR (glove)	SCOR (fasttext)
0.3204	0.3192	0.3182

SCOR (glove) means GloVe word embeddings (size=500) were used for retrieval

Conclusion

- The third generation tools are far superior in performance than the first and second generation tools
- SCOR semantic word embeddings are very generic:
  - They can be used to perform bug localization in those Java projects on word2vec was not trained on
  - Also, they are very effective for bug localization in non-Java projects
- It's time to merge techniques from 2nd and 3rd generations to develop hybrid approaches for IR-based bug localization

Thank you …

Questions?

Dataset will be made available at:

https://engineering.purdue.edu/RVL/Bugzbook/