SCOR: Source COde Retrieval With Semantics and Order

Shayan Akbar

Avi Kak

Purdue University

[An MSR 2019 Presentation]

Source Code Retrieval vs. Automatic Bug Localization

- Our primary interest is in addressing problems in source code retrieval.
- Automatic bug localization is convenient for testing such algorithms.
- Nonetheless it remains that not all our conclusions may apply to the more general problem of source code retrieval.

Contents

- What Have We Done --- in a Nutshell
- Word2vec and Contextual Semantics
- Markov Random Fields (MRF) for Imposing Order
- SCOR retrieval framework
- Results
- Conclusion

What Have We Done --- In a Nutshell

- In previous research, we have demonstrated how one can use **MRF-based modelling** to exploit term-term dependency constraints in source code retrieval.

- Speaking loosely, using term-term constraints means that you not only want to match queries with files on the basis of the frequencies of the individual terms, but also on the basis of the frequencies of pairs of terms taken together.

What Have We Done --- In a Nutshell (Contd.)

- Now that we know how to exploit term-term ordering constraints in retrieval, is it possible to further improve a retrieval framework with semantics --- at least with contextual semantics for now?

- **Contextual semantics means that we consider two terms similar if their contextual neighborhoods are similar.**

- **Word2vec** has emerged as a powerful (and popular) neural-network based algorithm for establishing contextual similarity between the terms.

What Have We Done --- In a Nutshell (Contd.)

- Our new work combines the ordering power of MRF with the power of contextual semantics you get from word2vec in a single unified retrieval framework we call **SCOR**

- **SCOR has outperformed the BoW and BoW+MRF based retrieval engines.**

- The performance improvements we obtain with **SCOR** are statistically significant but we are not setting the world on fire --- **AT LEAST NOT YET**.

What Exactly Does Word2vec Do?

- To get a sense of what is achieved by word2vec, suppose you analyze all of Shakespeare with word2vec, it would tell you that the words “king”, “queen”, “court”, “kingdom” are semantically related.

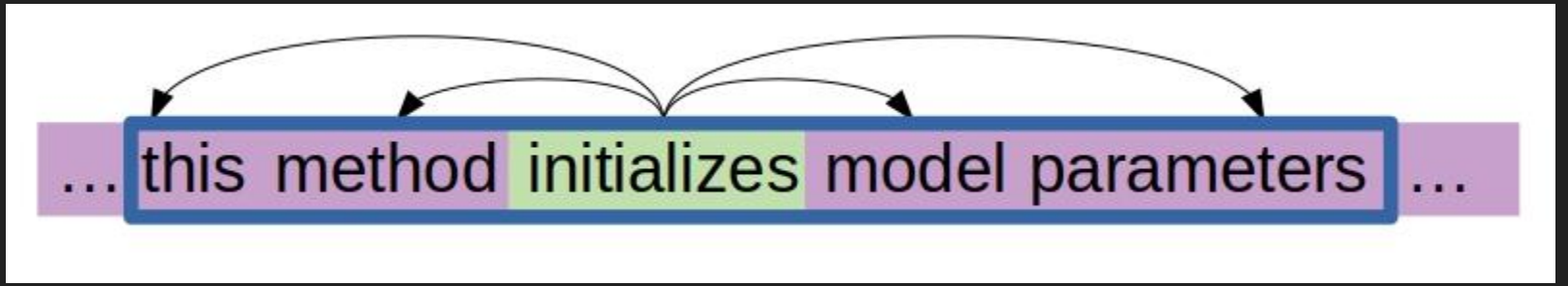
- **Word2vec represents each term by a numeric vector and the “semantic” similarity of two different words is related to the cosine distance between the two vectors.**

- The numeric vector for a term is referred to as its “semantic embedding”.

The word2vec neural-network model

- A single-layer neural network produces a vector space that holds contextually semantic relationships between words.
- Source code files are scanned using a window to generate pairs of target terms and context terms.

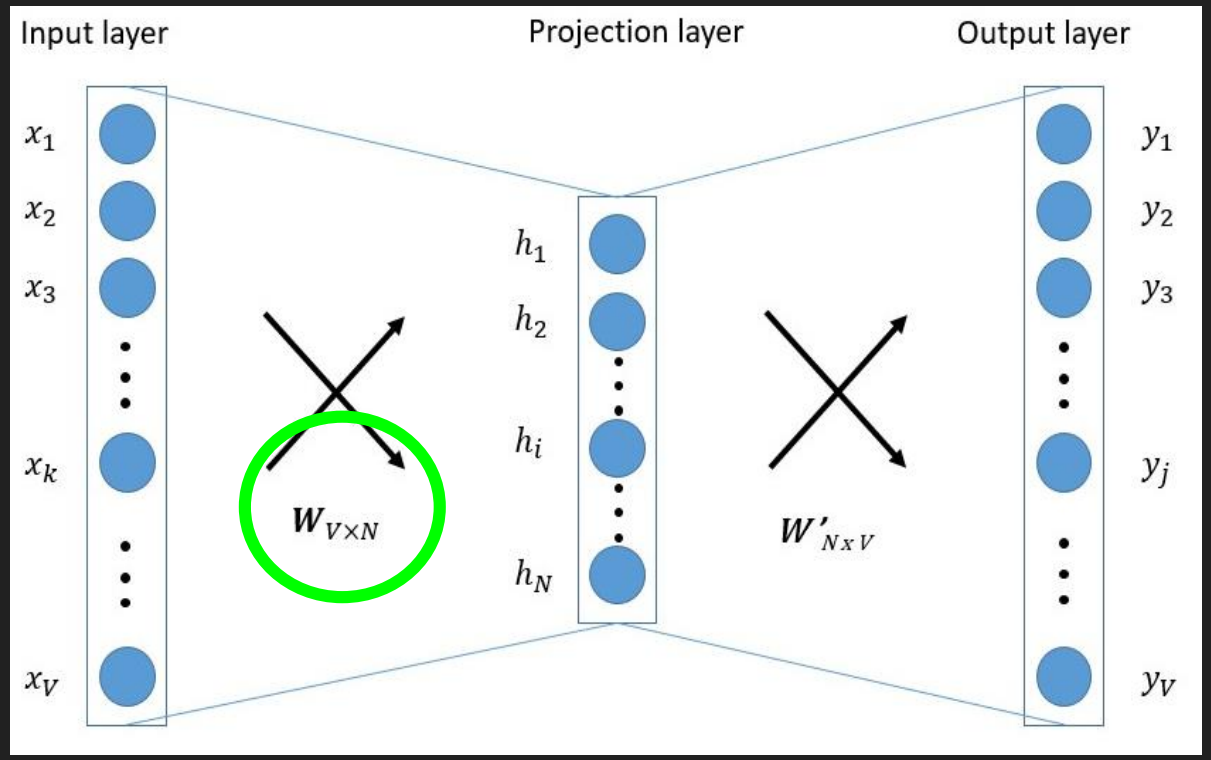
Training pairs: Target and Context terms



A window around the **target term "initialize"** with
four context terms **"this", "method", "model", and "parameters"**

Word2vec Neural Network

One-hot encoding of target term is provided at input



Softmax probabilities of context terms are computed at output.

After training finishes, each row of the weight matrix $\boldsymbol{W}_{V\times N}$ represents N-dim vector v_t for a word t in the vocabulary of size V.

But how to set N --- the size of the middle layer in the word2vec NN

- In the word2vec NN, the size of the middle layer determines the size of the numeric vectors you use for representing the individual software terms.
- When N is either too small or too large, the word embeddings lose their discriminative power.
- This question is answered by creating a semantic-similarity benchmark for experimenting with different values for N.
- This is what has been done in the past for NLP and for the medical domain, **but had not yet been attempted for the world of software.**

SoftwarePairs-400 --- A semantic similarity benchmark for the software world

- We believe that our SoftwarePairs-400 is the first semantic similarity benchmark for the software world.

- It contains a list of 400 software terms along with their commonly used abbreviations in programming.

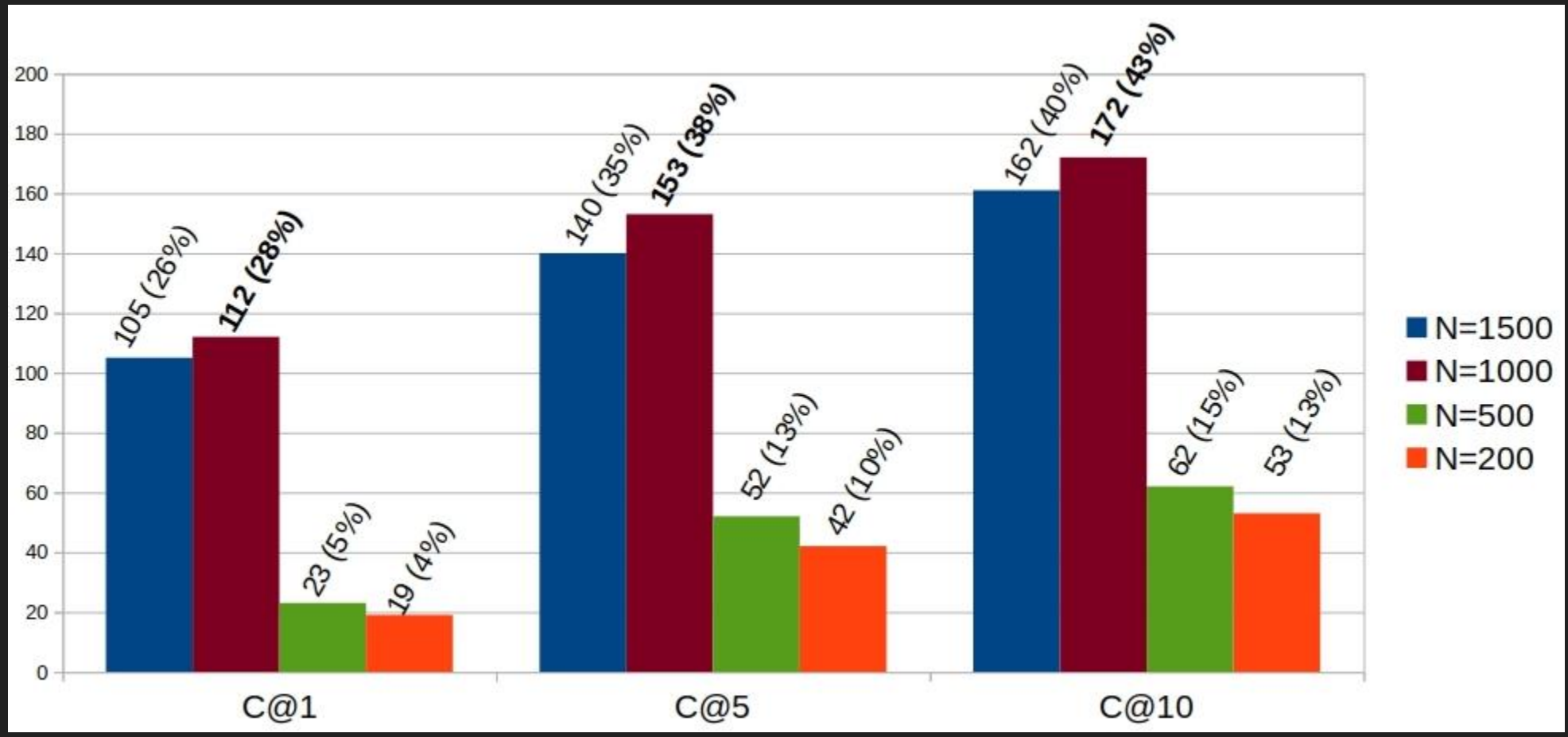
Sample pairs from SoftwarePairs-400

Software term	Abbreviation
delete	del
temporary	tmp
right	rt
minimum	min
number	num
median	med
column	col
total	tot
allocate	alloc

C@r: The metric used for finding the best vector size to use for word2vec

- **Correct at Rank r :** When we compare a vector for a software term against the vectors for all other terms in the benchmark list, how often does the correct abbreviation appear among **r** top-most ranks?

C@r results for SoftwarePairs-400 for different N

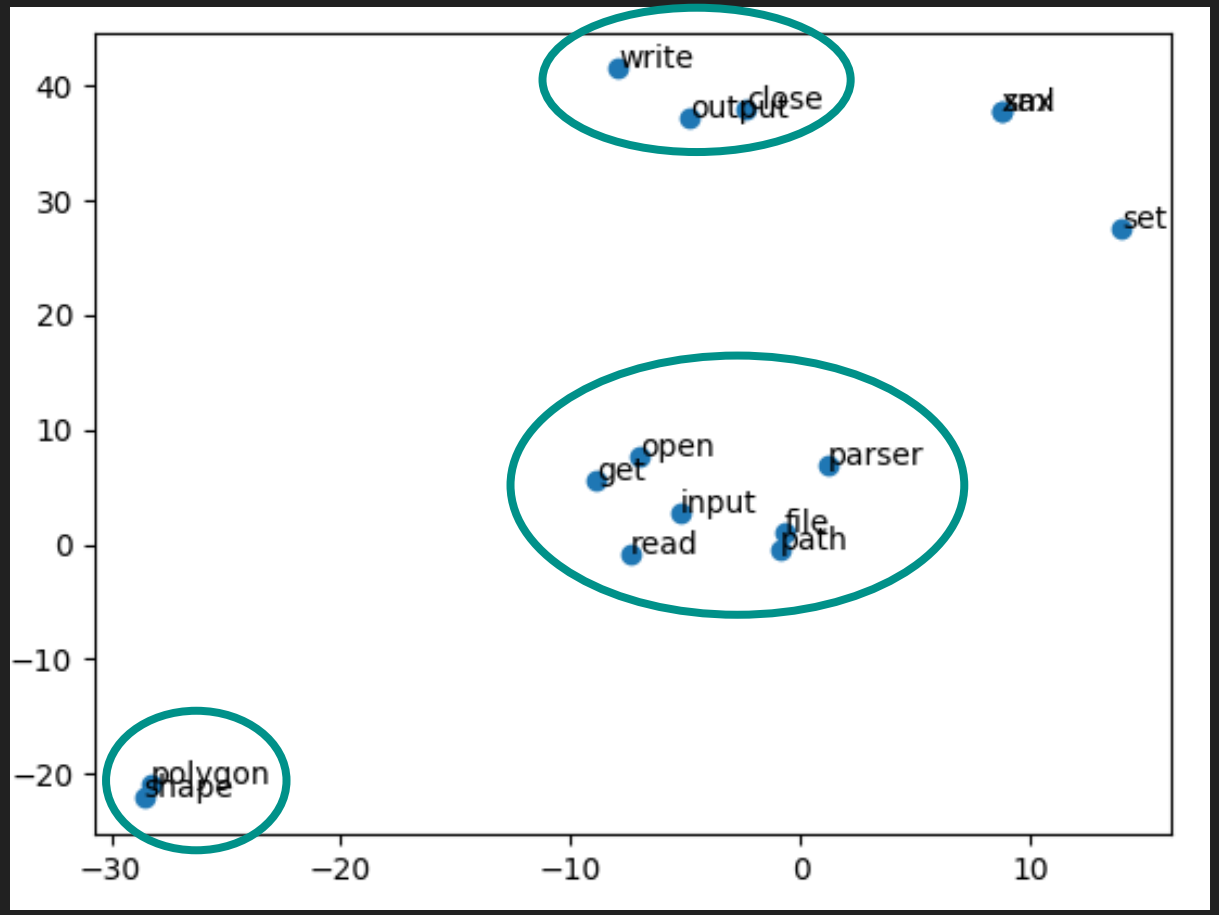


Getting Back to Our Main Business: How did we create semantic embeddings for software terms?

- **We downloaded 35000 Java repositories from GitHub.**
- We then constructed the word vectors (the embeddings) for 0.5 million software-centric terms from 1 billion word tokens.
- SCOR Word Embeddings can be found online at:

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

Some sample results from our database of software-centric semantic embeddings



Some words along with their top-3 semantically most similar words as discovered by word2vec

alexnet	delete	rotation	add	parameter
resnet	remove	angle	list	param
lenet	update	rot	set	method
imagenet	copy	lhornang	create	argument

Combining MRF based modelling with the semantic embeddings

- MRF gives us a way to model term-term relationships on the basis of order.
- Word2vec gives us a way to model term-term relationships on the basis of contextual semantics.
- **Can we combine the two and create a new class of retrieval engines?**

Markov Random Fields (MRF)

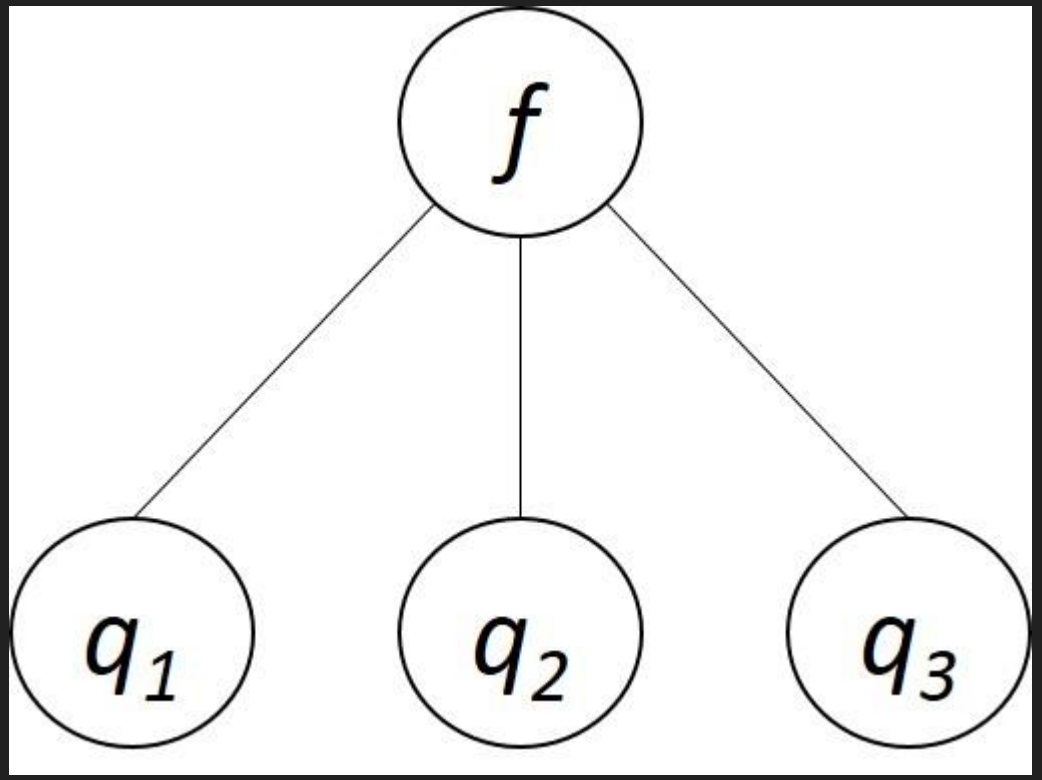
- MRF is an undirected graph *G* whose nodes satisfy the Markov property.
  - Nodes represent variables (for file f and bug report terms Q)
  - Arcs represent probabilistic dependencies between nodes

- MRF gives us the liberty to choose different kinds of probabilistic dependencies we want to encode in the retrieval model.

Two Dependency Assumptions of MRF

1. Full Independence Assumption
2. Sequential Dependence Assumption

The Full Independence (FI) Assumption

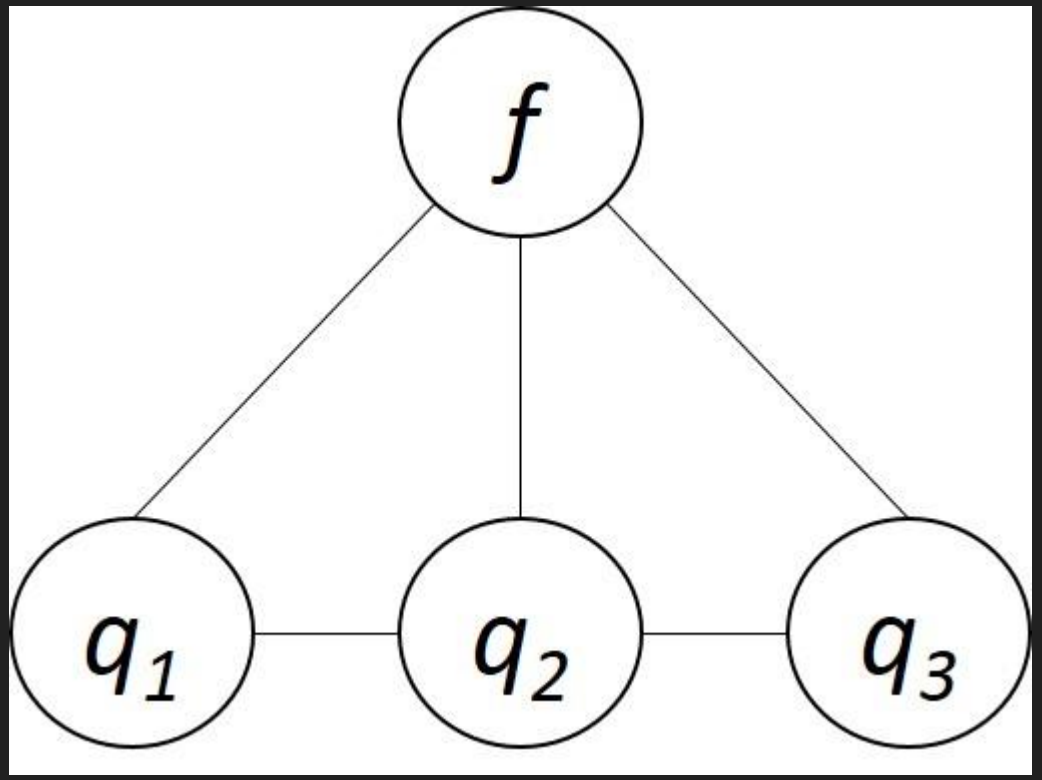


- f : file node
- q_i : query term nodes
- No arcs between query term nodes
- No dependency between query terms
- Same as a BoW model
- Relevance score computed as:

$$score_{fi}(f, Q) \propto tf(q_i, f)$$

$tf(q_i, f)$: frequency of query term q_i in f

The Sequential Dependence (SD) Assumption



- Same set of nodes
- Notice arcs between query term nodes
- Relevance score computed as:

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

$tf(q_i q_{i+1}, f)$: frequency of pair of query terms $q_i q_{i+1}$ in f

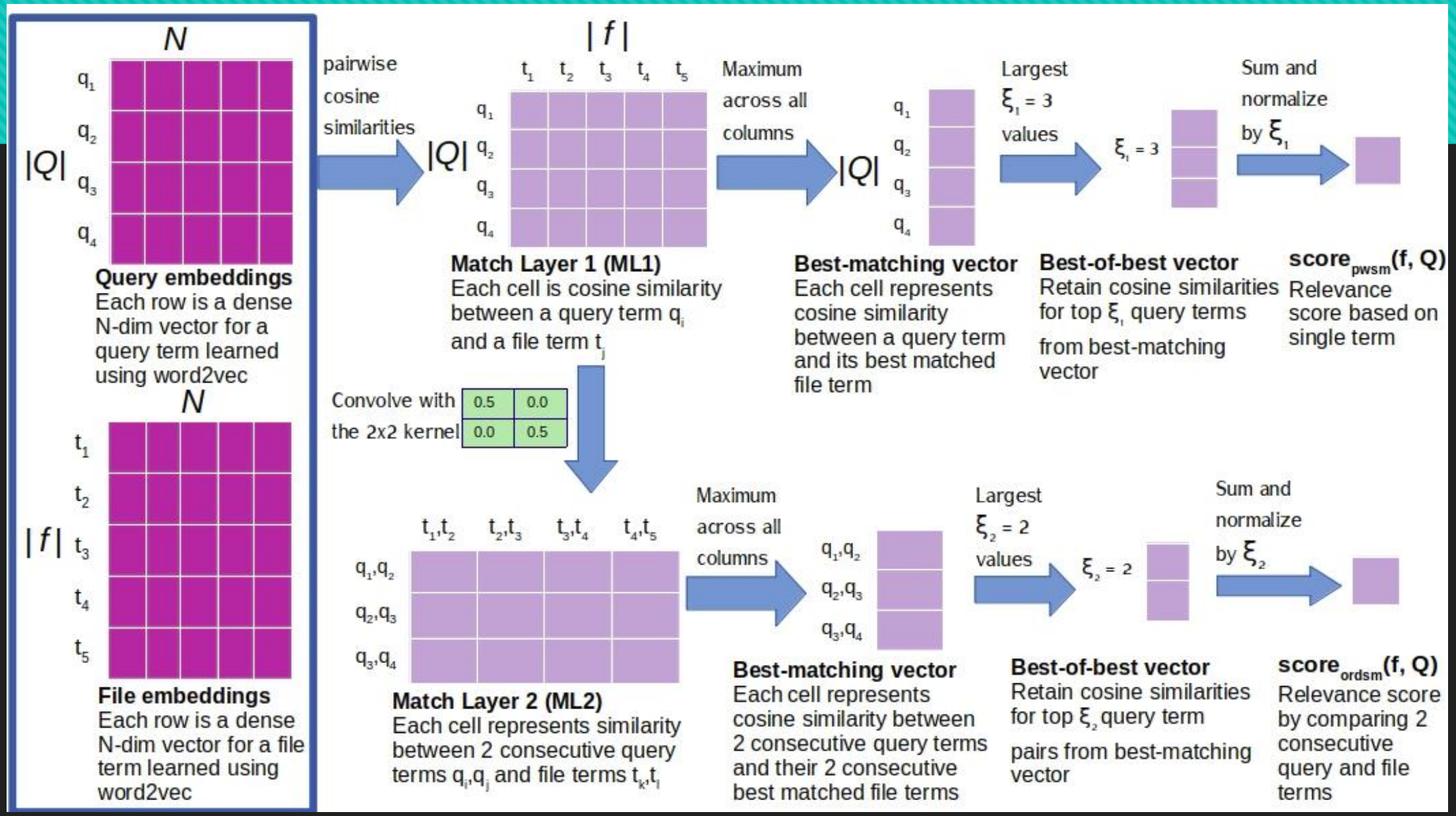
SCOR Retrieval Framework

SCOR: MRF + Semantic Embeddings

The SCOR retrieval engine produces two different relevance scores computed using word2vec based vectors:.

1. Per-Word Semantic Model
2. Ordered Semantic Model

SCOR architecture



SCOR --- Computing a Composite Score for a Repository File

Combine all measures of relevancy of a file to a query to create a composite relevance score using a **weighted aggregation of the score:**

$$score_{scor}(f, Q) = \alpha \,.\, score_{fi}(f, Q) + \beta \,.\, score_{sd}(f, Q) + \gamma \,.\, score_{pwsm}(f, Q) + \eta \,.\, score_{ordsm}(f, Q)$$

Results

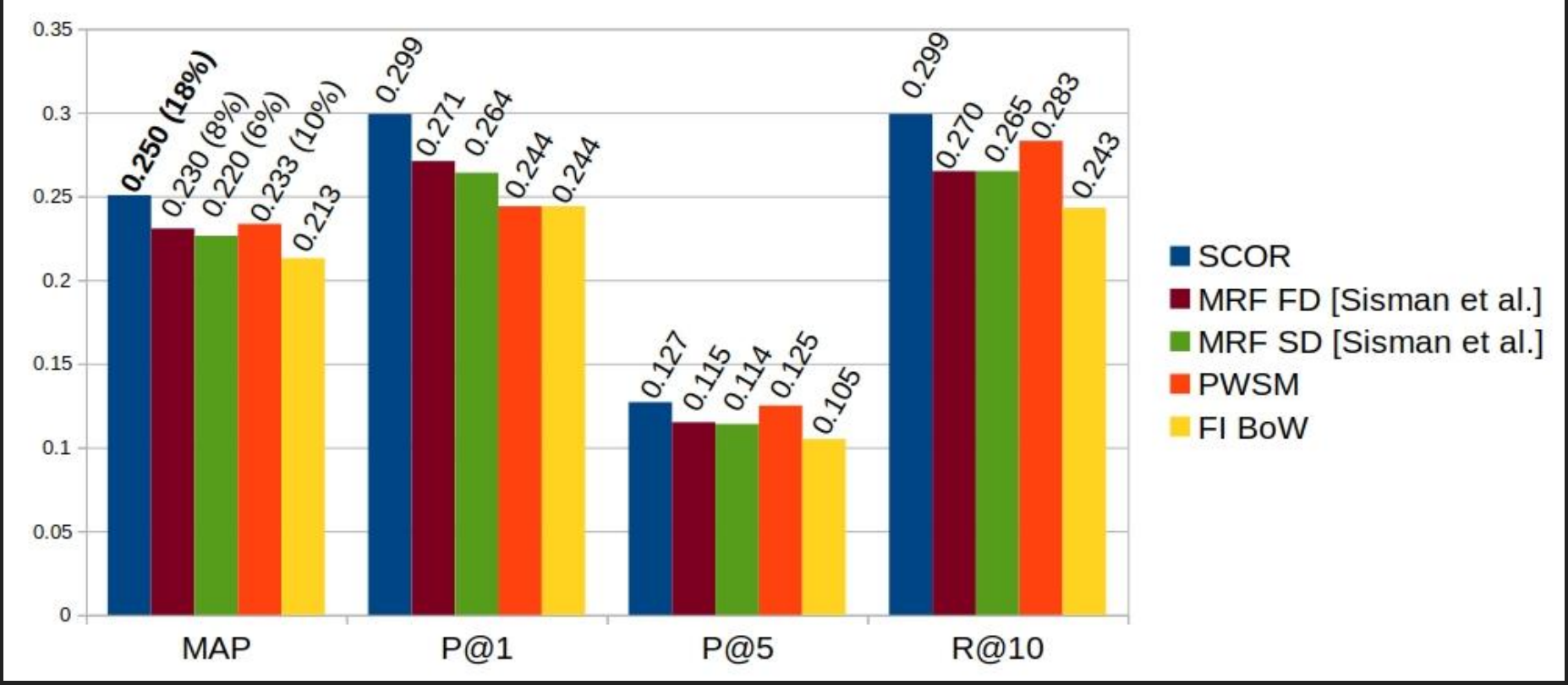
Highlights of the results

- Results on two popular datasets:
  1. Eclipse bug report queries taken from BUGLinks dataset
  2. AspectJ bug report queries taken from iBUGS dataset
- Experiments under two settings:
  1. With title of bug reports only
  2. With title as well as description of bug reports

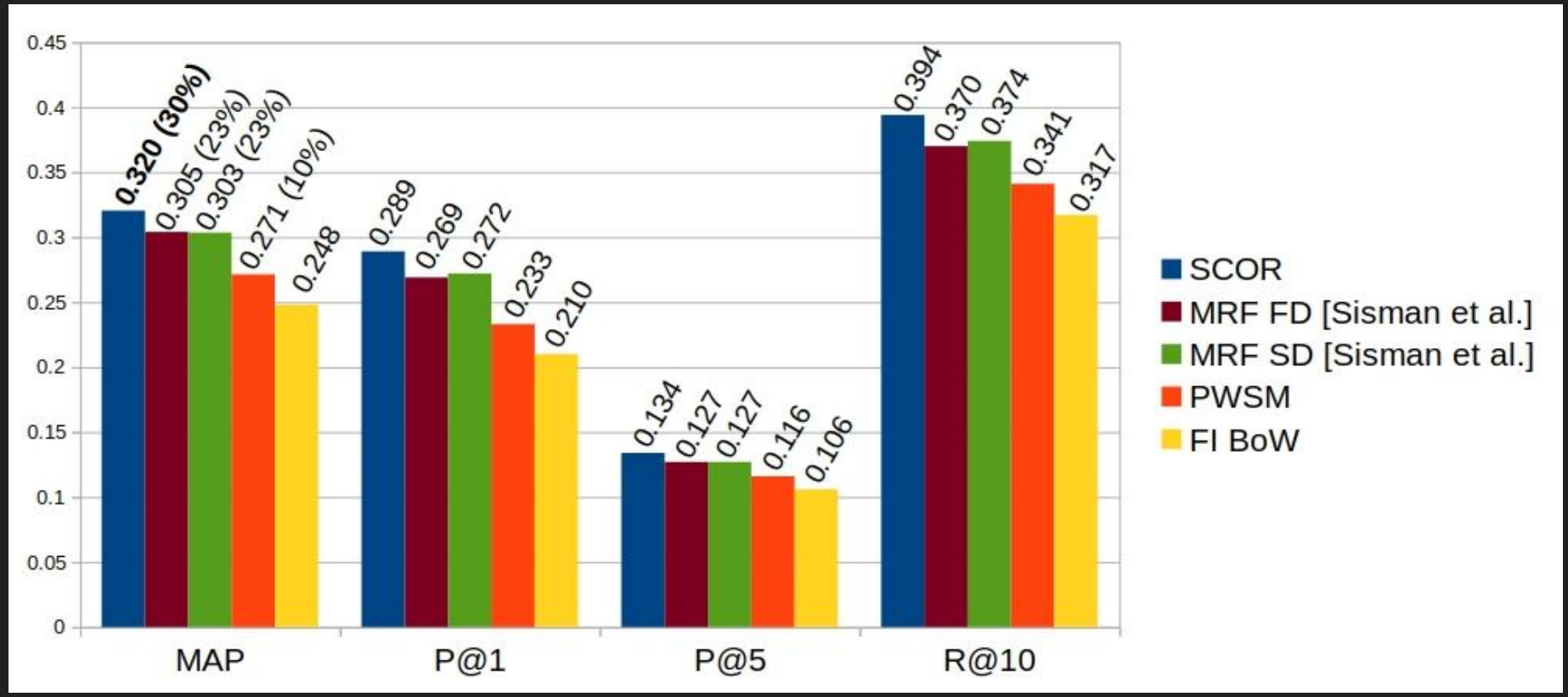
Highlights of the results (Contd.)

- SCOR outperforms FI BoW models with improvements in range 7% to 45%.
- SCOR outperforms pure MRF with improvements in range 6% to 30%.
- SCOR also outperforms BugLocator, BLUiR, and SCP-QR.
- SCOR word embeddings are sufficiently generic to be applied to new software repositories.

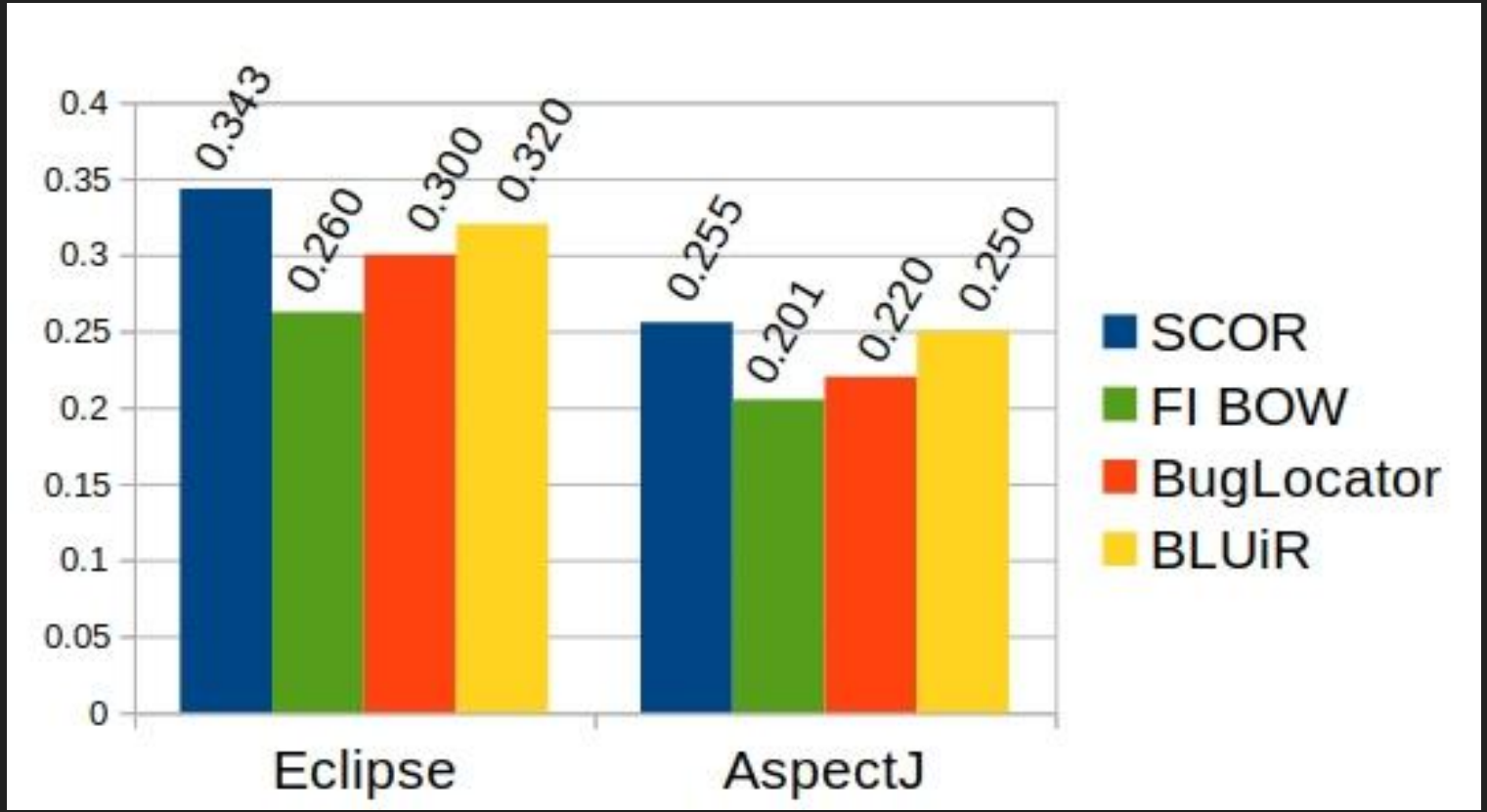
Results on 300 AspectJ “title+desc” bug reports



Results on 4000 Eclipse “title+desc” bug reports



Comparison with various BOW models
(not in MSR presentation)



~3000 queries ~300 queries

Conclusion

- SCOR is a novel retrieval framework that combines MRF and word2vec to model order and semantics together.
- SCOR gives state-of-the-art results on source code retrieval task of bug localization.
- In the process of developing SCOR we also generated semantic word embeddings for 0.5 million software-centric terms from 35000 Java repositories.

Thank you …

Questions?

SCOR Word Embeddings are available online and for download at:

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