Associating Natural Language Comment and Source Code Entities

The University of Texas at Austin

Code and data available at: http://bit.ly/NLCodeAssociations

Sheena Panthaplackel[1](spantha@cs.utexas.edu), Milos Gligoric[2], Raymond J. Mooney[1], Junyi Jessy Li[3]
Department of Computer Science[1], Department of Electrical and Computer Engineering[2], Department of Linguistics[3]
The University of Texas at Austin

Problem

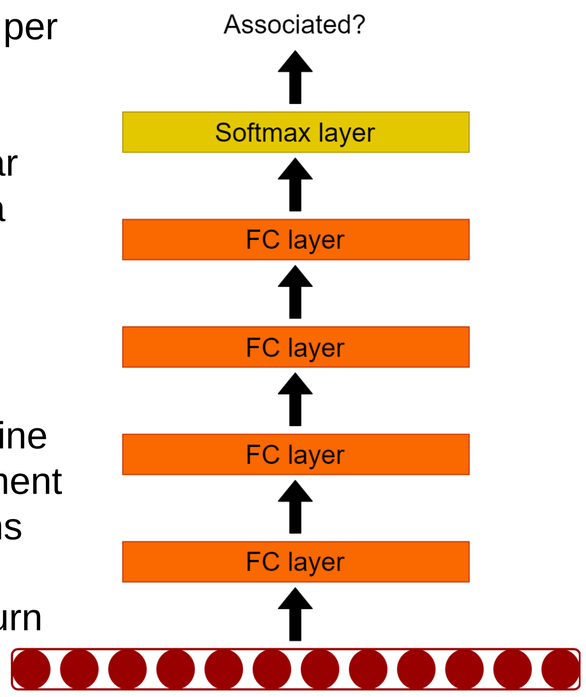
Motivation: Learning to relate comment and code elements is critical to automated systems for generating comments and code and detecting inconsistent comments and code.

```
/* @return Snapshot or null when there are problems reading it. */
public ConfigRepo.Snapshot getLatestConfig() {
   if (latestConfig == null) {
       try {
          updateConfigSnapshot();
        } catch (InterruptedException e) {
         Thread.currentThread().interrupt();
       }
   }
   return latestConfig;
}
```

Task: Given a noun phrase (NP) in a comment, we classify the relationship between the NP and each candidate code token in the corresponding method as either associated or not associated.

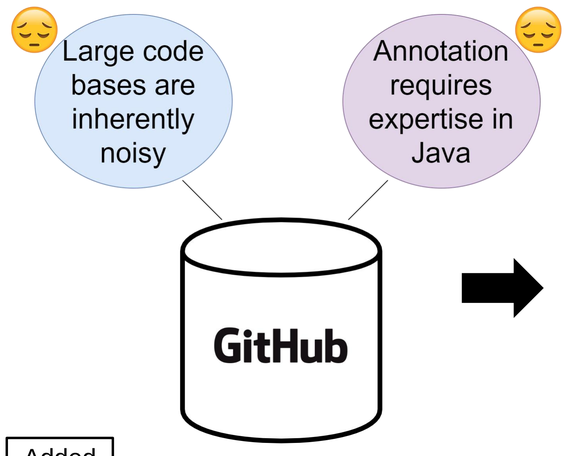
Features + Model

- Independent classification per candidate code token
- Feature vector captures:
  - Code structure**:** grammar and API relevant to Java
  - Comment and code context**:** averaged pre-trained embeddings
  - Relationship between comment and code: cosine similarity between comment and code representations
  - Lexical characteristics: overlap with NP and return statement tokens



Noisy Supervision

We propose obtaining noisy labels for this task by isolating parts of the comment and code that are edited at the same time, using GitHub's commit history feature.



Diff

```
 /**
- * @return the opcode of the next bytecode
+ * @return the opcode of the current bytecode
  */
 public int next() {
+   final int opcode = currentBC();
    setBCI(_nextBCI);
-   return currentBC();
+   return opcode;
 }
```

Added

```
+ * @return the opcode of the current bytecode
 public int next() {
+   final int opcode = currentBC();
    setBCI(_nextBCI);
+   return opcode;
 }
```

Deleted

```
- * @return the opcode of the next bytecode
 public int next() {
    setBCI(_nextBCI);
-   return currentBC();
 }
```

Primary Dataset

NP: Added NP in comment
Code token labels:
+ Added tokens
- Unchanged tokens (unlikely associated with NP that did not exist before)

Deletions Dataset

NP: Deleted NP in comment
Code token labels:
+ Deleted tokens
- Unchanged tokens (could be associated with NP that did exist before) *[MORE NOISE!]*

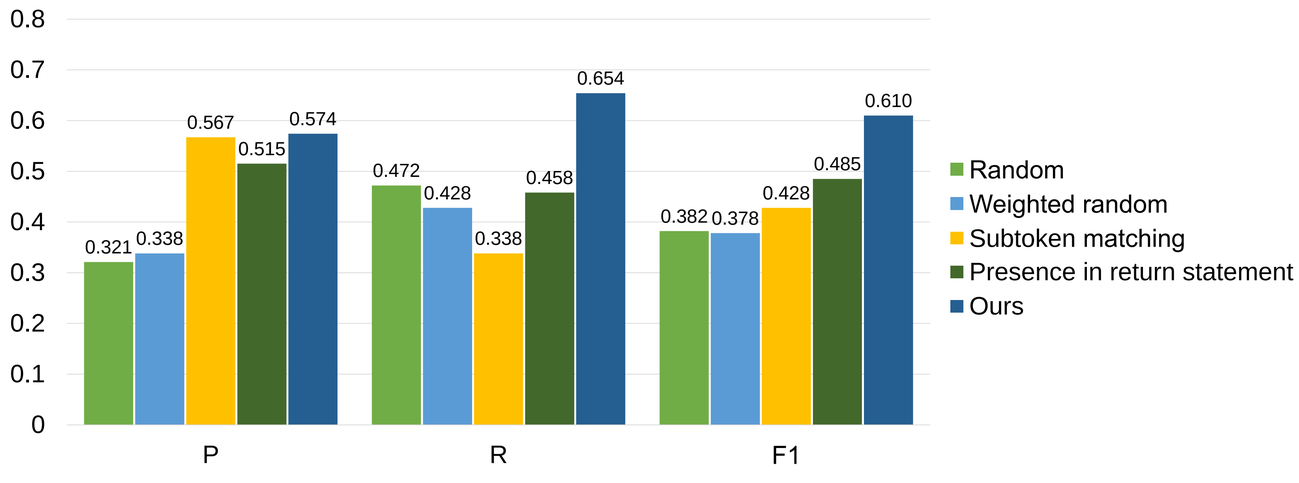
We mine simultaneous comment/code updates from ~1,000 Java projects on GitHub, and one of the authors manually annotated a test set by re-labeling positive labels in the primary dataset.

		Candidate Code Tokens		
Partition	**Examples**	**Total**	**Unique**	**Average**
Train	776	23,188	5,908	29.9
Validation	77	2,488	911	32.3
Test (annotated)	117	3,592	1,266	30.7
Deletions	867	25,203	6,186	29.1

Results

Conclusions:
- Learning from noisy supervision, our system outperforms multiple baselines
- "More noisy" data provides a valuable training signal, improving the scope for collecting data and training models

Comparing to rule-based baselines



Augmenting training with varying numbers of "more noisy" examples from deletions