# Supplementary Material for Structural Language Models of Code

| | Java | C# |
|---|---|---|
| #projects - training | 9 | 25 |
| #projects - validation | 1 | 2 |
| #projects - test | 1 | 3 |
| #examples - training | 1,309,842 | 16,295 |
| #examples - validation | 10,000 | 8,183 |
| #examples - test | 20,000 | 3,305 |
| Avg. number of paths | 27.8 | 131.1 |
| Avg. source length - lines | 10.4 | 57.5 |
| Avg. source length - tokens | 77.7 | 264.3 |
| Avg. source length - subtokens | 100.6 | 343.6 |
| Avg. target length - tokens | 5.4 | 3.9 |
| Avg. target length - subtokens | 7.8 | 5.0 |
| Avg. target length - tree nodes | 3.8 | 3.9 |
| Avg. target length - tree targets | 10.8 | 10.8 |

*Figure 1.* Statistics of our datasets. When not mentioned otherwise, the statistic was measured on the training set.

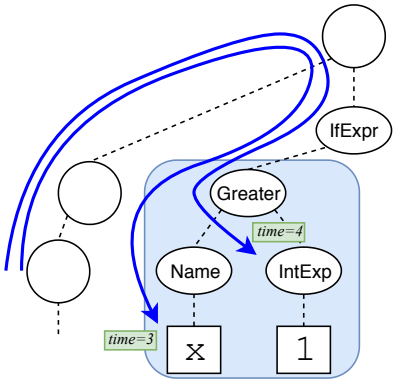


*Figure 2.* Efficient computation: partial paths for different time steps share the same prefix, allowing a shared computation. In this example, the prefix is the shared path from the leaf (not shown) to `Greater`, and is much longer than either of the suffixes.

## 1. Data Statistics

Figure 1 shows some statistics of our used datasets. In Java: for the validation set, we randomly sampled $10,000$ examples from the raw validation set; for the test set, we randomly sampled $20,000$ examples from the raw test set.

We will release all datasets, raw and preprocessed, with the final version.

## 2. Additional Evaluation Details

For both Java and C# models, we experimented with the following hyper-parameter values. We performed beam search on the validation set after every training iteration, and we selected the best configuration and checkpoint according to accuracy@1 on the validation set. After the best configuration was chosen, we ran a single evaluation run on the test set.

- $\overrightarrow{f} \in \{LSTM, Transformer\}$ – how to encode each path.
- LSTM #layers $\in \{1, 2\}$
- $d_{subtoken} \in \{256, 512\}$ – embedding size.
- Transformer layers $\in \{0, 1, 2, 3, 4\}$
- $lr \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ – learning rate
- Learning rate decay every $\{10000, 20000, 40000\}$ steps.

| Model | Exact-match (acc@k) | | One SubToken Diff | | One Token Diff | | Tree@k | |
|---|---|---|---|---|---|---|---|---|
| | @1 | @5 | @1 | @5 | @1 | @5 | @1 | @5 |
| Transformer$_{\text{base}}$ +copy | 16.65 | 24.05 | 23.08 | 34.06 | 29.39 | 43.46 | 34.68 | 50.52 |
| BiLSTM→LSTM +copy | 16.93 | 23.17 | 22.39 | 31.68 | 27.23 | 38.92 | 34.29 | 49.72 |
| seq2tree +copy | 16.81 | 23.04 | 24.02 | 33.89 | 32.67 | 43.75 | 38.14 | 52.36 |
| SLM (this work) | **18.04** | **24.83** | **24.40** | **35.19** | **33.68** | **46.57** | **39.10** | **55.32** |

*Table 1.* Examining the gap between *acc@k* and *tree@k*: the *acc@k* and *tree@k* results here are the same as in Table 1 in the paper; *One SubToken Diff* allows a single *sub*token mismatch; *One Token Diff* allows a single token mismatch.

## 3. Qualitative Analysis cont. - Correct Tree, Incorrect Names

In Section 7 of the paper we discuss the gap between *acc@k* and *tree@k*. We find that 30% of the examples in the gap could have been *exact match* if a single subtoken prediction was fixed; 74% of the examples in the gap could have been *exact match* if a single identifier prediction was fixed. Table 1 shows the accuracy of our model and the leading baselines if a single subtoken or a single token mismatches were counted as correct: *One SubToken Diff* and *One Token Diff* are similar to *exact match*, except that they allow a single subtoken or a single token mistake, respectively. As Table 1 shows, not only that our model performs better than the baselines in *exact match*, it also shows a greater potential for improvement.

## 4. Copying Single Subtokens

In addition to scoring the entire token to be copied, we also score each of the subtokens composing it according to their position. For each position $i$, we add a scoring function $s_{copy_i}$, such that $s_{copy_i}(\ell)$ produces the copying score of the $i$'th subtoken of $\ell$, which we denote as $\ell_i$:

$$s_w = s_{\text{gen}}(w) + \sum_{\text{val}(\ell)=w} s_{\text{copy\_token}}(\ell) + \sum_i \sum_{\text{val}(\ell_i)=w} s_{\text{copy}_i}(\ell)$$

$$Pr(a|\mathcal{S}) = \text{softmax}(s)$$

Where $s_{\text{copy\_token}}$ is the scoring function of copying the entire token, described in Section 3.3 in the paper.

For example, a token of `getX` is scored entirely using $s_{\text{copy\_token}}$; each of its subtokens, `get` and `X`, are scored using $s_{copy_1}$ and $s_{copy_2}$ respectively. That is, the model can either copy the entire token, or copy only some of its subtokens. This ability is especially useful in generating a name like `setX`, where `getX` appears in the context, and `X` is any unknown, user-defined, subtoken; the model learns to generate `set` from the vocabulary, and copy only the subtoken `X`.

## 5. Example: Usefulness of Copy Mechanism

As shown in Section 6 of the paper, the ability to copy is crucial for the any-code completion task, because of the repetitive use of identifiers and symbols in programs. Figure 3 shows a representative example for the necessity of the copy mechanism: generating the ground truth `zkfcUgi.getShortUserName()` is feasible *only* thanks to the copy mechanism, since `zkfc` is obviously an UNK subtoken which was not observed in the training data.

In this case, since both `zkfcUgi` and `getShortUserName` appear in context, both were copied as *entire tokens*, rather than generated using subtokens. This example also shows how the ability to copy *entire tokens* ease the generation process by reducing the number of target symbols (our SLM model is able to copy and combine single subtokens as well).

## 6. Java Examples

Figures 4 to 13 contain examples from our test set for the any-code completion task in Java, along with the prediction of our model and some of the baselines. The highlighted expressions are the true references that should be generated. Indentation and line breaks may have been altered for typesetting reasons.

```
protected void checkRpcAdminAccess() throws IOException, AccessControlException {
  UserGroupInformation ugi = UserGroupInformation.getCurrentUser();
  UserGroupInformation zkfcUgi = UserGroupInformation.getLoginUser();
  if (adminAcl.isUserAllowed(ugi)
    || ugi.getShortUserName().equals( zkfcUgi.getShortUserName() )) {
      LOG.info("Allowed RPC access from " + ugi
        + " at " + Server.getRemoteAddress());
      return;
    }
  String msg = "Disallowed RPC access from " + ugi
    + " at " + Server.getRemoteAddress() + ". Not listed in " + DFSConfigKeys.DFS_ADMIN;
  LOG.warn(msg);
  throw new AccessControlException(msg);
}
```

| | | | |
|---|---|---|---|
| True ref: | `zkfcUgi.getShortUserName()` | | |
| SLM top-5 candidates: | **`zkfcUgi.getShortUserName()`** | (11.7%) | (exact match) |
| | `DFSConfigKeys.DFS` | (4.5%) | |
| | `zkfcUgi.getUserName()` | (2.6%) | (tree-match) |
| | `zkfcUgi.getUser()` | (1.7%) | (tree-match) |
| | `zkfcUgi.getUserId()` | (0.6%) | (tree-match) |

*Entirely copied tokens* are marked in brown; unknown *copied subtokens* are marked in blue; *in-vocabulary* subtokens are marked in black; subtokens that are *both in-vocabulary and copied* from context are marked in purple.

*Figure 3.* A Java Any-Code Completion example from our test set along with the predictions of our model. The predictions of the baselines are shown in Figure 7 below.

## 7. C# Examples

Figures 14 to 21 contain examples from our test set for the restricted completion task in C# along with the prediction of our model some of the baselines. The highlighted expressions are the true references that should be generated. Indentation and line breaks may have been altered for typesetting reasons.

```
private C findCounter(T key) {
    int i = key.ordinal();
    if (counters[i] == null) {
        counters[i] = newCounter(key);
    }
    return (C) counters[i];
}
```

| Model | Prediction |
|---|---|
| True ref: | `(C) counters[i]` |
| SLM (this work) | **`(C) counters[i]`** (71.6%)<br>`(C) this` (6.3%)<br>`counters[i]` (4.8%) |
| Transformer$_{\text{base}}$ +copy | `(C) this`<br>**`(C) counters[i]`**<br>`(C) counters` |
| BiLSTM→LSTM +copy | `(C) this`<br>**`(C) counters[i]`**<br>`counters[i]` |
| Seq2tree +copy | **`(C) counters[i]`**<br>`(C) counters[i].ordinal()`<br>`(C) counters.get(i)` |

```
private void handleTaskFinishedEvent(TaskFinishedEvent event) {
  TaskInfo taskInfo = info.tasksMap.get(event.getTaskId());
  taskInfo.counters = event.getCounters();
  taskInfo.finishTime = event.getFinishTime();
  taskInfo.status = TaskStatus.State.SUCCEEDED.toString();
  taskInfo.successfulAttemptId = event.getSuccessfulTaskAttemptId();
}
```

| Model | Prediction |
|---|---|
| True ref: | `event.getTaskId()` |
| SLM (this work) | `event.getTaskName()` (8.8%)<br>`event.getId()` (8.2%)<br>`event.getTask()` (3.4%)<br>`event.getName()` (3.3%)<br>**`event.getTaskId()`** (3.3%) |
| Transformer$_{\text{base}}$ +copy | `event.getTaskInfo()`<br>**`event.getTaskId()`**<br>`event.getId()`<br>`event.getTask()`<br>`taskInfo.getTaskId()()` |
| BiLSTM→LSTM +copy | `event.name`<br>`event.type`<br>`event.getId()`<br>`event.id`<br>`event.getKey()` |
| Seq2tree +copy | `event.getId()`<br>`event.getPath()`<br>`event.getDescription()`<br>`event.getTaskName()`<br>`event.getTaskName(` (*Syntax error*) |

*Figure 4.* Java examples from our test set along with the predictions of our model and the baselines.

```
private static void log(String value) {
    if (value!= null && value.length() > 55)
        value = value.substring(0, 55) + "...";
    LOG.info(value);
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `value.length() > 55` | |
| SLM (this work) | `value.length() > 0` | (9.6%) |
| | **`value.length() > 55`** | (7.3%) |
| | `value.startsWith("...")` | (1.8%) |
| Transformer$_{\text{base}}$ +copy | **`value.length() > 55`** | |
| | `value.length() > 0)` | |
| | `value.length() > 1` | |
| BiLSTM→LSTM +copy | **`value.length() > 55`** | |
| | `value.startsWith("")` | |
| | `value.startsWith("...")` | |
| Seq2tree +copy | `value.length() 55` | (*Syntax error*) |
| | `value.endsWith("info")` | |
| | `value.length() 55` | (*Syntax error*) |

```
private List<INode> initChildren() {
  if (children == null) {
    final ChildrenDiff combined = new ChildrenDiff();
    for (DirectoryDiff d = DirectoryDiff.this; d != null; d = d.getPosterior()) {
      combined.combinePosterior(d.diff, null);
    }
    children = combined.apply2Current(ReadOnlyList.Util.asList(
        currentDir.getChildrenList(Snapshot.CURRENT_STATE_ID)));
  }
  return children;
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `d = d.getPosterior()` | |
| SLM (this work) | `d = d.getParent()` | (18.8%) |
| | `d = d.getChildrenList()` | (14.9%) |
| | `d = d` | (4.5%) |
| | `d = combined` | (2.5%) |
| | **`d = d.getPosterior()`** | (1.8%) |
| Transformer$_{\text{base}}$ +copy | `d = d` | |
| | `d = d.diff` | |
| | `d = d.getChildren()` | |
| | `d = d.currentDir` | |
| | `d = d.currentStateId` | |
| BiLSTM→LSTM +copy | `--d` | |
| | `d = d` | |
| | `d = d.getParent()` | |
| | `d = d.next` | |
| | `d = d.get()` | |
| Seq2tree +copy | `d d.next` | (*Syntax error*) |
| | `d d.parent` | (*Syntax error*) |
| | `d d.getParent()` | (*Syntax error*) |
| | `d d.getChildren()` | (*Syntax error*) |
| | `d d.getRoot()` | (*Syntax error*) |

*Figure 5.* Java examples from our test set along with the predictions of our model and the baselines.

```
public float getProgress() {
    this.readLock.lock();
    try {
        if (this.currentAttempt != null) {
            return this.currentAttempt.getProgress();
        }
        return 0;
    } finally {
      this.readLock.unlock();
    }
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `this.currentAttempt.getProgress()` | |
| SLM (this work) | `this.currentAttempt.getCount()` | (31.3%) |
| | `-1` | (30.6%) |
| | `this.currentAttempt.get()` | (1.5%) |
| | `this.currentAttempt.getTime()` | (1.2%) |
| | **`this.currentAttempt.getProgress()`** | (0.9%) |
| Transformer$_{\text{base}}$ +copy | **`this.currentAttempt.getProgress()`** | |
| | `this.currentAttempt.floatValue()` | |
| | `this.currentAttempt.getFloat()` | |
| | `this.currentAttempt.get()` | |
| | `this.currentAttempt.getTime()` | |
| BiLSTM→LSTM +copy | **`this.currentAttempt.getProgress()`** | |
| | `this.currentAttempt.float()` | |
| | `this.currentAttempt.get()` | |
| | `this.currentAttempt.size()` | |
| | `this.currentAttempt.compute()` | |
| Seq2tree +copy | **`this.currentAttempt.getProgress()`** | |
| | `this.currentAttempt.floatValue()` | |
| | `this.currentAttempt.get()` | |
| | `this.currentAttempt.getValue()` | |
| | `(float)this.currentAttempt.size()` | |

```
public int compareTo(LongWritable o) {
    long thisValue = this.value;
    long thatValue = o.value;
    return (thisValue < thatValue ? -1 : (thisValue == thatValue ? 0 : 1));
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `thisValue == thatValue ? 0 : 1` | |
| SLM (this work) | `thisValue == thisValue ? 0 : 1` | (16.3%) |
| | **`thisValue == thatValue ? 0 : 1`** | (11.0%) |
| | `thisValue == value ? 0 : 1` | (9.5%) |
| Transformer$_{\text{base}}$ +copy | `thatValue >> thatValue` | |
| | `thatValue > thatValue ? 1 : 0` | |
| | `thatValue > thatValue` | |
| BiLSTM→LSTM +copy | `thisValue - thatValue` | |
| | `thatValue & thatValue` | |
| | `thatValue ? 1 : 0` | |
| Seq2tree +copy | `thisValue thatValue` | (*Syntax error*) |
| | `thisValue thatValue 0 1` | (*Syntax error*) |
| | `thisValue thatValue 1 0` | (*Syntax error*) |

*Figure 6.* Java examples from our test set along with the predictions of our model and the baselines.

```
private static String getNameServiceId(
    Configuration conf, String addressKey) {
  String nameserviceId = conf.get(DFS_NAMESERVICE_ID);
  if (nameserviceId != null) {
    return nameserviceId;
  }
  Collection<String> nsIds = getNameServiceIds(conf);
  if (1 == nsIds.size()) {
    return nsIds.toArray(new String[1])[0];
  }
  String nnId = conf.get(DFS_HA_NAMENODE_ID_KEY);
  return
    getSuffixIDs(conf, addressKey, null, nnId, LOCAL_ADDRESS_MATCHER)[0];
}
```

| Model | Predictions | |
|---|---|---|
| True ref: | `nsIds.size()` | |
| SLM (this work) | **`nsIds.size()`** | (83.7%) |
| | `conf.size()` | (3.0%) |
| | `getSuffixIDs(conf).length` | (2.5%) |
| Transformer$_{\text{base}}$ +copy | `-1` | |
| | `ns.size()` | |
| | `conf.size()` | |
| BiLSTM→LSTM +copy | `-1` | |
| | `Integer.MAX_VALUE` | |
| | `conf.size()` | |
| Seq2tree +copy | `1` | |
| | **`nsIds.size()`** | |
| | `stringPool.blank` | |

```
protected void checkRpcAdminAccess() throws
    IOException, AccessControlException {
  UserGroupInformation ugi = UserGroupInformation.getCurrentUser();
  UserGroupInformation zkfcUgi = UserGroupInformation.getLoginUser();
  if (adminAcl.isUserAllowed(ugi) ||
    ugi.getShortUserName().equals(zkfcUgi.getShortUserName())) {
      LOG.info("Allowed RPC access from " + ugi
        + " at " + Server.getRemoteAddress());
      return;
    }
  String msg = "Disallowed RPC access from " + ugi
    + " at " + Server.getRemoteAddress()
    + ". Not listed in " + DFSConfigKeys.DFS_ADMIN;
  LOG.warn(msg);
  throw new AccessControlException(msg);
}
```

| Model | Predictions | |
|---|---|---|
| True ref: | `zkfcUgi.getShortUserName()` | |
| SLM (this work) | **`zkfcUgi.getShortUserName()`** | (11.7%) |
| | `DFSConfigKeys.DFS` | (4.5%) |
| | `zkfcUgi.getUserName()` | (2.6%) |
| Transformer$_{\text{base}}$ +copy | `server.getRemoteAddress()` | |
| | `server.getRemoteUserName()` | |
| | `server.getShortUserName()` | |
| BiLSTM→LSTM +copy | `server.getUserName()` | |
| | `zkfcUgi.getUserName()` | |
| | `ugiUgi.getUserName()` | |
| Seq2tree +copy | `dfsConfigKeys.dfsAdmin` | |
| | `zkfc.getUserName()` | |
| | `zkfcUgi.getRemoteAddress()` | |

*Figure 7.* Java examples from our test set along with the predictions of our model and the baselines.

```
static String replaceSubstitution(
    String base, Pattern from, String to, boolean repeat) {
  Matcher match = from.matcher(base);
  if (repeat) {
    return match.replaceAll(to);
  } else {
    return match.replaceFirst(to);
  }
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `match.replaceAll(to)` | |
| SLM (this work) | `match.toString()`<br>**`match.replaceAll(to)`**<br>`match.replaceAll(to, from)` | (9.0%)<br>(8.2%)<br>(6.5%) |
| Transformer$_{base}$ +copy | `match.replaceFirst(to)`<br>`replace.replaceFirst(to)`<br>`matcher.replaceFirst(to)` | |
| BiLSTM→LSTM +copy | `match.getFirst()`<br>`match.replaceFirst(to)`<br>`match.replaceFirst(to, to)` | |
| Seq2tree +copy | `match.replaceFirst(base)`<br>`match.replaceFirst(to)`<br>`match.replaceFirst(repeat)` | |

```
public void responseReceived(ResponseReceivedEvent event) {
  RequestResult result = event.getRequestResult();
  Date startDate = result.getStartDate();
  Date stopDate = result.getStopDate();
  long elapsed = stopDate.getTime() - startDate.getTime();
  synchronized (this) {
    this.lastE2Elatency = elapsed;
  }
  if (LOG.isDebugEnabled()) {
    int statusCode = result.getStatusCode();
    String etag = result.getEtag();
    HttpURLConnection urlConnection =
        (HttpURLConnection) event.getConnectionObject();
    int contentLength = urlConnection.getContentLength();
    String requestMethod = urlConnection.getRequestMethod();
    long threadId = Thread.currentThread().getId();
    LOG.debug(String.format(
      "SelfThrottlingIntercept:: ResponseReceived:
      ... threadId=%d, Status=%d, Elapsed(ms)=%d,
      ... ETAG=%s, contentLength=%d, requestMethod=%s",
      threadId, statusCode, elapsed, etag, contentLength, requestMethod));
  }
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `LOG.isDebugEnabled()` | |
| SLM (this work) | `elapsed != null`<br>**`LOG.isDebugEnabled()`**<br>`!LOG.isDebugEnabled()` | (32.1%)<br>(29.0%)<br>(2.4%) |
| Transformer$_{base}$ +copy | `stopDate != null`<br>`result.hasStatusCode()`<br>`result.hasStatusCode() != elapsed` | |
| BiLSTM→LSTM +copy | `result != null`<br>`elapsed > 0`<br>`result.getStatusCode() == workflowConstants.STATUS` | |
| Seq2tree +copy | `event.getConnectionObject() instanceof HttpUrlConnection`<br>`startDate != null`<br>**`LOG.isDebugEnabled()`** | |

*Figure 8.* Java examples from our test set along with the predictions of our model and the baselines.

```
private static boolean isNameResolved(InetAddress address) {
  String hostname = address.getHostName();
  String ip = address.getHostAddress();
  return !hostname.equals(ip) || NetUtils.isLocalAddress(address);
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `address.getHostName()` | |
| SLM (this work) | **`address.getHostname()`** | (3.5%) |
| | **`address.getHostName()`** | (2.0%) |
| | `inetAddress.getByName(address.getAddress())` | (0.7%) |
| Transformer$_{base}$ +copy | `address.getHostAddress()` | |
| | `address.getLastElement().getValue()` | |
| | `address.getAddress()` | |
| BiLSTM→LSTM +copy | `address.getHostAddress()` | |
| | `address.getPort()` | |
| | `address.getAddress()` | |
| Seq2tree +copy | `address.getHostAddress()` | |
| | `address.getPort()` | |
| | `address.getAddress()` | |

```
private synchronized void initJournals(List<URI> dirs) {
  int minimumRedundantJournals = conf.getInt(
      DFSConfigKeys.DFS_NAMENODE_EDITS_DIR_MINIMUM_KEY,
      DFSConfigKeys.DFS_NAMENODE_EDITS_DIR_MINIMUM_DEFAULT);
  journalSet = new JournalSet(minimumRedundantJournals);
  for (URI u : dirs) {
    boolean required =
        FSNamesystem.getRequiredNamespaceEditsDirs(conf).contains(u);
    if (u.getScheme().equals(NNStorage.LOCAL_URI_SCHEME)) {
      StorageDirectory sd = storage.getStorageDirectory(u);
      if (sd != null) {
        journalSet.add(
            new FileJournalManager(conf, sd, storage),
            required, sharedEditsDirs.contains(u));
      }
    } else {
      journalSet.add(createJournal(u),
          required, sharedEditsDirs.contains(u));
    }
  }
  if (journalSet.isEmpty()) {
    LOG.error("No edits directories configured!");
  }
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `u.getScheme()` | |
| SLM (this work) | `u.getName()` | (27.4%) |
| | **`u.getScheme()`** | (13.1%) |
| | `u.getVersion()` | (8.2%) |
| Transformer$_{base}$ +copy | `journalSet.LOCAL_URI_SCHEME` | |
| | `u.getName()` | |
| | `Boolean.true` | |
| BiLSTM→LSTM +copy | `u.toString()` | |
| | `Boolean.true` | |
| | `u.getURI()` | |
| Seq2tree +copy | **`u.getScheme()`** | |
| | `u.getName()` | |
| | `storage.getLocalUriScheme()` | |

*Figure 9.* Java examples from our test set along with the predictions of our model and the baselines.

```
static EnumSet<FileAttribute> parse(String s) {
  if (s == null || s.length() == 0) {
    return EnumSet.allOf(FileAttribute.class);
  }
  EnumSet<FileAttribute> set = EnumSet.noneOf(FileAttribute.class);
  FileAttribute[] attributes = values();
  for (char c : s.toCharArray() ) {
    int i = 0;
    for (; i < attributes.length && c != attributes[i].symbol; i++) ;
    if (i < attributes.length) {
      if (!set.contains(attributes[i])) {
        set.add(attributes[i]);
      } else {
        throw new IllegalArgumentException("There are more than one '"
            + attributes[i].symbol + "' in " + s);
      }
    } else {
      throw new IllegalArgumentException("'" + c + "' in "
          + s + " is undefined.");
    }
  }
  return set;
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `s.toCharArray()` | |
| SLM (this work) | **`s.toCharArray()`** | (22.4%) |
| | `attributes[0].value` | (18.5%) |
| | `attributes[undefined].length` | (4.6%) |
| Transformer$_{base}$ +copy | `s.split(" "` | |
| | `set.split(" ")` | |
| | `attributes.keySet()` | |
| BiLSTM→LSTM +copy | `attributes.length` | |
| | `attributes[0]` | |
| | `attributes[0].next` | |
| Seq2tree +copy | `set.toArray()` | |
| | **`s.toCharArray()`** | |
| | `set.toCharArray()` | |

```
public static Path[] stat2Paths(FileStatus[] stats) {
  if (stats == null)
    return null;
  Path[] ret = new Path[stats.length];
  for (int i = 0; i < stats.length; ++i) {
    ret[i] = stats[i].getPath() ;
  }
  return ret;
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `stats[i].getPath()` | |
| SLM (this work) | **`stats[i].getPath()`** | (25.2%) |
| | `Path(stats[i])` | (3.3%) |
| | `new Path(stats[i], charset)` | (2.5%) |
| Transformer$_{base}$ +copy | `stats[i]` | |
| | **`stats[i].getPath()`** | |
| | `new Path(stats[i])` | |
| BiLSTM→LSTM +copy | `stats[i]` | |
| | `new Path(stats[i])` | |
| | `stats[i].toString()` | |
| Seq2tree +copy | `stats[i]` | |
| | `new Path(stats[i])` | |
| | `stat(stats[i])` | |

*Figure 10.* Java examples from our test set along with the predictions of our model and the baselines.

```
void ensureCurrentDirExists() throws IOException {
  for (
      Iterator<StorageDirectory> it = storage.dirIterator();
      it.hasNext(); ) {
    StorageDirectory sd = it.next();
    File curDir = sd.getCurrentDir();
    if (!curDir.exists() && !curDir.mkdirs()) {
      throw new IOException("Could not create directory " + curDir);
    }
  }
}
```

| Model | Prediction |
|---|---|
| True ref: | `!curDir.exists()` |
| SLM (this work) | **`!curDir.exists()`** (29.0%)<br>`curDir != null` (25.8%)<br>`curDir.exists()` (24.4%) |
| Transformer$_{\text{base}}$ +copy | `curDir != null`<br>**`!curDir.exists()`**<br>`curDir.exists()` |
| BiLSTM→LSTM +copy | `curDir != null`<br>`curDir.exists()`<br>`sd != null` |
| Seq2tree +copy | `curDir != null`<br>`curDir.exists()`<br>**`!curDir.exists()`** |

```
public static byte[] getXAttr(final Map<?, ?> json, final String name)
    throws IOException {
  if (json == null) {
    return null;
  }
  Map<String, byte[]> xAttrs = toXAttrs(json);
  if (xAttrs != null) {
    return xAttrs.get(name);
  }
  return null;
}
```

| Model | Prediction |
|---|---|
| True ref: | `xAttrs.get(name)` |
| SLM (this work) | **`xAttrs.get(name)`** (28.2%)<br>`xAttrs.get(xAttrs)` (5.8%)<br>`xAttrs.toByteArray()` (4.4%) |
| Transformer$_{\text{base}}$ +copy | **`xAttrs.get(name)`**<br>`xAttrs.toByteArray()`<br>`new byte[0]` |
| BiLSTM→LSTM +copy | `xAttrs.getBytes()`<br>`new byte[0]`<br>`xAttrs.toByteArray()` |
| Seq2tree +copy | **`xAttrs.get(name)`**<br>`xAttrs.get()`<br>`xAttrs.get(0)` |

*Figure 11.* Java examples from our test set along with the predictions of our model and the baselines.

```
private void setFlag(long flag) {
  long prev;
  do {
    prev = unsafe.getLongVolatile(null, this.slotAddress);
    if ((prev & flag) != 0) {
      return;
    }
  } while (!unsafe.compareAndSwapLong(
              null, this.slotAddress, prev, prev | flag));
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `(prev & flag)` | |
| SLM (this work) | **`(prev & flag)`** | (8.9%) |
| | `(prev & flagSlot)` | (5.4%) |
| | `unsafe.get(prev)` | (5.0%) |
| Transformer$_{\text{base}}$ +copy | **`(prev & flag)`** | |
| | `(prev \| flag)` | |
| | `unsafe.compareTo(prev)` | |
| BiLSTM→LSTM +copy | `prev` | |
| | `prev + 1` | |
| | `prev - 1` | |
| Seq2tree +copy | `unsafe prev flag` | (*Syntax error*) |
| | `(volatile prev unsafe.get())` | (*Syntax error*) |
| | `(volatile prev unsafe.getLongVolatile(null, prev))` | (*Syntax error*) |

```
public synchronized void setInput(byte[] b, int off, int len) {
  if (b == null) {
    throw new NullPointerException();
  }
  if (off < 0 || len < 0 || off > b.length - len) {
    throw new ArrayIndexOutOfBoundsException();
  }
  finished = false;
  if (len > uncompressedDirectBuf.remaining()) {
    this.userBuf = b;
    this.userBufOff = off;
    this.userBufLen = len;
  } else {
    ((ByteBuffer) uncompressedDirectBuf).put(b, off, len);
    uncompressedDirectBufLen = uncompressedDirectBuf.position();
  }
  bytesRead += len;
}
```

| Model | Predictions | |
|---|---|---|
| True ref: | `len < 0` | |
| SLM (this work) | **`len < 0`** | (41.3%) |
| | `off > b.length` | (23.4%) |
| | `len > b.length` | (14.1%) |
| Transformer$_{\text{base}}$ +copy | `off < 0` | |
| | **`len < 0`** | |
| | `b == null` | |
| BiLSTM→LSTM +copy | `off < 0` | |
| | **`len < 0`** | |
| | `b == null` | |
| Seq2tree +copy | `off < 0` | |
| | **`len < 0`** | |
| | `0 < off` | |

*Figure 12.* Java examples from our test set along with the predictions of our model and the baselines.

```
private int readData(byte[] buf, int off, int len) throws IOException {
  int bytesRead = 0;
  while (bytesRead < len) {
    int n = IOUtils.wrappedReadForCompressedData(
         in, buf, off + bytesRead, len - bytesRead);
    if (n < 0) {
      return bytesRead;
    }
    bytesRead += n;
  }
  return len;
}
```

| Model | Prediction |
|---|---|
| True ref: | `off + bytesRead` |
| SLM (this work) | `bytesRead - bytesRead` (35.0%)<br>**`off + bytesRead`** (14.1%)<br>`off - bytesRead` (9.4%) |
| Transformer$_{\text{base}}$ +copy | `off - bytesRead`<br>`off + len`<br>`len - bytesRead` |
| BiLSTM→LSTM +copy | `-bytesRead`<br>`bytesRead++`<br>`bytesRead - bytesRead` |
| Seq2tree +copy | `compressed bytesRead` (*Syntax error*)<br>**`off + bytesRead`**<br>`len - bytesRead` |

```
private Path getPath(int curId, int limitPerDir, Type type) {
  if (curId <= 0) {
    return basePath;
  }
  String name = "";
  switch(type) {
    case FILE:
      name = FILE_PREFIX + new Integer(curId % limitPerDir).toString();
      break;
    case DIRECTORY:
      name = DIR_PREFIX + new Integer(curId % limitPerDir).toString();
      break;
  }
  Path base = getPath((curId / limitPerDir), limitPerDir, Type.DIRECTORY);
  return  new Path(base, name);
}
```

| Model | Prediction |
|---|---|
| True ref: | `new Path(base, name)` |
| SLM (this work) | **`new Path(base, name)`** (6.0%)<br>`new Path(base, name, limitPerDir)` (2.9%)<br>`new Path(base, name, type)` (2.8%) |
| Transformer$_{\text{base}}$ +copy | `new Path(base)`<br>`new Path(name)`<br>`getPath(base)` |
| BiLSTM→LSTM +copy | `new Path(base)`<br>`new File(base)`<br>`new Path(base.getPath())` |
| Seq2tree +copy | `new Path(base)`<br>`new File(base, name)`<br>**`new Path(base, name)`** |

*Figure 13.* Java examples from our test set along with the predictions of our model and the baselines.

```
private static IEnumerable<Token> OfSequence(
    this IEnumerable<Token> tokens, Token nameToken, TypeDescriptor info)
{
  var nameIndex = tokens.IndexOf(t => t.Equals(nameToken));
  if (nameIndex >= 0)
  {
    return info.NextValue.MapValueOrDefault(
        _ => info.MaxItems.MapValueOrDefault(
          n => tokens.Skip(nameIndex + 1).Take(n),
               tokens.Skip(nameIndex + 1).TakeWhile(v => v.IsValue())),
        tokens.Skip(nameIndex + 1).TakeWhile(v => v.IsValue()));
  }
  return new Token[] { };
}
```

| Model | Prediction |
|---|---|
| True ref: | `nameIndex >= 0` |
| SLM (this work) | **`nameIndex >= 0`** (22.6%)<br>`nameIndex == -1` (19.1%)<br>`nameIndex > -1` (13.9%) |
| BiLSTM→LSTM +copy | `!nameIndex`<br>`nameIndex == -1`<br>`nameIndex < 0` |
| *GNN*→$\mathcal{NAG}$ (Brockschmidt et al., 2019) | `nameIndex == 0`<br>`nameIndex > 0`<br>`nameIndex < 0` |

```
public static IEnumerable<T[]> Group<T>(
    this IEnumerable<T> source, int groupSize)
{
  if (groupSize < 1)
  {
    throw new ArgumentOutOfRangeException(nameof(groupSize));
  }
  T[] group = new T[groupSize];
  int groupIndex = 0;
  foreach (var item in source)
  {
    group[groupIndex++] = item;
    if (groupIndex == groupSize)
    {
      yield return group;
      group = new T[groupSize];
      groupIndex = 0;
    }
  }
}
```

| Model | Prediction |
|---|---|
| True ref: | `groupIndex == groupSize` |
| SLM (this work) | `groupIndex < 0` (21.4%)<br>`groupIndex == -1` (10.3%)<br>`groupIndex < groupIndex` (5.3%) |
| BiLSTM→LSTM +copy | `group.IsNullOrEmpty()`<br>`groupGroup[groupIndex++]`<br>`group.EndsWith(group)` |
| *GNN*→$\mathcal{NAG}$ (Brockschmidt et al., 2019) | `groupIndex == 0`<br>`groupIndex == 1`<br>**`groupIndex == groupSize`** |

*Figure 14.* C# examples from our test set of the restricted completion task along with the predictions of our model and the baselines.

```
internal static void AddLine(StringBuilder builder,
    string value, int maximumLength)
{
  if (builder == null)
  {
    throw new ArgumentNullException(nameof(builder));
  }
  if (value == null)
  {
    throw new ArgumentNullException(nameof(value));
  }
  if (maximumLength < 1)
  {
    throw new ArgumentOutOfRangeException(nameof(value));
  }

  value =  value.Trim() ;

  builder.AppendWhen(builder.Length > 0, Environment.NewLine);
  do
  {
    var wordBuffer = 0;
    var words = value.Split(' ');
    for (var i = 0; i < words.Length; i++)
    {
      if (words[i].Length < (maximumLength - wordBuffer))
      {
        builder.Append(words[i]);
        wordBuffer += words[i].Length;
        if ((maximumLength - wordBuffer) > 1 && i != words.Length - 1)
        {
          builder.Append(" ");
          wordBuffer++;
        }
      }
      else if (words[i].Length >= maximumLength && wordBuffer == 0)
      {
        builder.Append(words[i].Substring(0, maximumLength));
        wordBuffer = maximumLength;
        break;
      }
      else break;
    }
    value = value.Substring(Math.Min(wordBuffer, value.Length));
    builder.AppendWhen(value.Length > 0, Environment.NewLine);
  }
  while (value.Length > maximumLength);
  builder.Append(value);
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `value.Trim()` | |
| SLM (this work) | **`value.Trim()`** | (16.0%) |
| | `value.Substring(0, maximumLength)` | (10.9%) |
| | `value.Replace(maximumLength, maximumLength` | (10.7%) |
| BiLSTM→LSTM +copy | `maximumLength - 1` | |
| | **`value.Trim()`** | |
| | `valueLength++` | |
| *GNN→NAG* | `value + <UNK>` | |
| | `value + maximumLength` | |
| | `value.Substring(0, maximumLength)` | |

*Figure 15.* C# examples from our test set of the restricted completion task along with the predictions of our model and the baselines.

```
public static string[] TrimStringArray(this IEnumerable<string> array)
{
    return array.Select(item => item.Trim() ).ToArray();
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `item.Trim()` | |
| SLM (this work) | **`item.Trim()`** | (20.1%) |
| | `item.ToUpperInvariant()` | (3.5%) |
| | `item.ToUpper()` | (1.6%) |
| BiLSTM→LSTM +copy | **`item.Trim()`** | |
| | `item.ToTrim()` | |
| | `item.]` | (*Syntax error*) |
| *GNN*→$\mathcal{NAG}$ (Brockschmidt et al., 2019) | `item + <UNK>` | |
| | `item + item` | |
| | `item + 1` | |

```
public static string Camelize(this string input)
{
    var word = Pascalize(input);
    return word.Substring(0, 1).ToLower() + word.Substring(1);
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `word.Substring(0, 1)` | `word.Substring(1)` |
| SLM (this work) | **`word.Substring(0, 1)`** | **`word.Substring(1)`** |
| | `word.Trim()` | `wordData.Substring(1)` |
| | `word.Substring(1)` | `word.Substring(0, 1)` |
| BiLSTM→LSTM +copy | `input.Replace("&", " )` | `input.Replace("&", " <UNK> )` |
| | `input.Replace(1, '')` | `input + "." + input` |
| | `input.Replace("&", "")` | `input.Substring(0, 1)` |
| *GNN*→$\mathcal{NAG}$ | `word.CombineWith(<UNK>)` | `word.CombineWith(<UNK>)` |
| | `word.Trim()` | `word + <UNK>` |
| | `word.CombineWith(input)` | `word.Replace(<UNK>, <UNK>)` |

*Figure 16.* C# examples from our test set of the restricted completion task along with the predictions of our model and the baselines.

```
public string Truncate(string value, int length, string truncationString,
    TruncateFrom truncateFrom = TruncateFrom.Right)
{
  if (value == null)
    return null;

  if (value.Length == 0)
    return value;

  if (truncationString == null)
    truncationString = string.Empty;

  if (truncationString.Length > length)
    return truncateFrom == TruncateFrom.Right ?
      value.Substring(0, length) : value.Substring(value.Length - length);

  var alphaNumericalCharactersProcessed = 0;

  if (value.ToCharArray().Count(char.IsLetterOrDigit) <= length)
    return value;

  if (truncateFrom == TruncateFrom.Left)
  {
    for (var i = value.Length - 1; i > 0; i--)
    {
      if (char.IsLetterOrDigit(value[i]))
        alphaNumericalCharactersProcessed++;

      if (alphaNumericalCharactersProcessed + truncationString.Length
          == length)
        return truncationString + value.Substring(i);
    }
  }

  for (var i = 0; i < value.Length - truncationString.Length; i++)
  {
    if (char.IsLetterOrDigit(value[i]))
      alphaNumericalCharactersProcessed++ ;

    if (alphaNumericalCharactersProcessed + truncationString.Length
        == length)
      return value.Substring(0, i + 1) + truncationString;
  }

  return value;
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `alphaNumericalCharactersProcessed++` | |
| SLM (this work) | **`alphaNumericalCharactersProcessed++`** | (48.1%) |
| | `iCount++` | (5.8%) |
| | `iIndex++` | (1.6%) |
| BiLSTM→LSTM +copy | `i++` | |
| | `truncation++` | |
| | `alpha--` | |
| *GNN*→$\mathcal{NAG}$ | **`alphaNumericalCharactersProcessed++`** | |
| | `alphaNumericalCharactersProcessed--` | |
| | `--alphaNumericalCharactersProcessed` | |

*Figure 17.* C# examples from our test set of the restricted completion task along with the predictions of our model and the baselines.

```
public static int BinarySearch<TItem, TSearch>(
    this IList<TItem> list, TSearch value,
    Func<TSearch, TItem, int> comparer)
{
  if (list == null)
  {
    throw new ArgumentNullException("list");
  }
  if (comparer == null)
  {
    throw new ArgumentNullException("comparer");
  }

  var lower = 0;
  var upper = list.Count - 1;

  while (lower <= upper)
  {
    var middle = lower + (upper - lower) / 2;
    var comparisonResult = comparer(value, list[middle]);
    if ( comparisonResult < 0 )
    {
      upper = middle - 1;
    }
    else if ( comparisonResult > 0 )
    {
      lower = middle + 1;
    }
    else
    {
      return middle;
    }
  }

  return lower;
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `comparisonResult < 0` | `comparisonResult > 0` |
| SLM (this work) | **`comparisonResult < 0`**<br>`comparisonResult > 0`<br>`middle == comparisonResult` | **`comparisonResult > 0`**<br>`comparisonResult < 0`<br>`comparisonResult == 0` |
| BiLSTM→LSTM +copy | `lowerResult == middle`<br>`lowerResult == 0`<br>`lower != middle` | `lower < 0`<br>`lower + "."`<br>`lower != middle` |
| *GNN*→$\mathcal{NAG}$ | `comparisonResult == 0`<br>`comparisonResult > 0`<br>**`comparisonResult < 0`** | `comparisonResult == 0`<br>**`comparisonResult > 0`**<br>`comparisonResult == middle` |

*Figure 18.* C# examples from our test set of the restricted completion task along with the predictions of our model and the baselines.

```
public override string ToString()
{
  // use reflection to display all the properties that
  // ... have non default values
  StringBuilder result = new StringBuilder();
  var props = this.GetType().GetTypeInfo().DeclaredProperties;
  result.AppendLine("{");
  foreach (var prop in props)
  {
    if (prop.Name != "Content" && prop.Name != "Subtitle"
        && prop.Name != "Title" && prop.Name != "UniqueId")
    {
        object value = prop.GetValue(this);
        bool valueIsNull = value == null;
        object defaultValue = Common.GetDefault(prop.PropertyType);
        bool defaultValueIsNull = defaultValue == null;
        if ((valueIsNull != defaultValueIsNull)
            // one is null when the other isn't
          || (!valueIsNull
              && (value.ToString() != defaultValue.ToString())))
            // both aren't null, so compare as strings
        {
          result.AppendLine(prop.Name + " : " + prop.GetValue(this));
        }
    }
  }
  result.AppendLine("}");
  return result.ToString();
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `!valueIsNull` | |
| SLM (this work) | **`!valueIsNull`** | (52.4%) |
| | `!defaultValueIsNull` | (9.0%) |
| | `!valueIsNull.IsNullOrEmpty()` | (3.2%) |
| BiLSTM→LSTM +copy | `!defaultValueIsNull` | |
| | `(defaultValueIsNull \|\| value)` | |
| | `(defaultValueIsNull \|\| defaultValue)` | |
| *GNN*→$\mathcal{NAG}$ | **`!valueIsNull`** | |
| | `!defaultValueIsNull` | |
| | `!!valueIsNull` | |

*Figure 19.* C# examples from our test set of the restricted completion task along with the predictions of our model and the baselines.

```
public TradierOrderResponse PlaceOrder(string accountId,
    TradierOrderClass classification,
    TradierOrderDirection direction,
    string symbol,
    decimal quantity,
    decimal price = 0,
    decimal stop = 0,
    string optionSymbol = "",
    TradierOrderType type = TradierOrderType.Market,
    TradierOrderDuration duration = TradierOrderDuration.GTC)
{
    //Compose the request:
    var request = new RestRequest("accounts/{accountId}/orders");
    request.AddUrlSegment("accountId", accountId.ToString());

    //Add data:
    request.AddParameter("class", GetEnumDescription(classification));
    request.AddParameter("symbol", symbol);
    request.AddParameter("duration", GetEnumDescription(duration));
    request.AddParameter("type", GetEnumDescription(type));
    request.AddParameter("quantity", quantity);
    request.AddParameter("side", GetEnumDescription(direction));

    //Add optionals:
    if (price > 0) request.AddParameter("price", Math.Round(price, 2));
    if (stop > 0) request.AddParameter("stop", Math.Round(stop, 2));
    if (optionSymbol != "")
        request.AddParameter("option_symbol", optionSymbol);

    //Set Method:
    request.Method = Method.POST;

    return Execute<TradierOrderResponse>(request,
        TradierApiRequestType.Orders);
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `optionSymbol != ""` | |
| SLM (this work) | **`optionSymbol != ""`** | (5.5%) |
| | `optionSymbol == ""` | (4.4%) |
| | `optionSymbol.IsNullOrEmpty()` | (1.1%) |
| BiLSTM→LSTM +copy | `!stopSymbol` | |
| | `stopSymbol != optionSymbol` | |
| | `(stopSymbol " && optionSymbol)` | (*Syntax error*) |
| *GNN→NAG* | `optionSymbol == <UNK>` | |
| | `optionSymbol == symbol` | |
| | `optionSymbol != symbol` | |

*Figure 20.* C# examples from our test set of the restricted completion task along with the predictions of our model and the baselines.

```
[Test, TestCaseSource("GetLeanDataLineTestParameters")]
public void GetSourceMatchesGenerateZipFilePath(
    LeanDataLineTestParameters parameters)
{
    var source = parameters.Data.GetSource(
        parameters.Config, parameters.Data.Time.Date, false);
    var normalizedSourcePath = new FileInfo(source.Source).FullName;
    var zipFilePath = LeanData.GenerateZipFilePath(
        Globals.DataFolder, parameters.Data.Symbol,
        parameters.Data.Time.Date,
        parameters.Resolution, parameters.TickType);
    var normalizeZipFilePath = new FileInfo(zipFilePath).FullName;
    var indexOfHash = normalizedSourcePath.LastIndexOf(
        "#", StringComparison.Ordinal);
    if (indexOfHash > 0)
    {
        normalizedSourcePath =
            normalizedSourcePath.Substring(0, indexOfHash);
    }
    Assert.AreEqual(normalizeZipFilePath, normalizedSourcePath);
}
```

| Model | Prediction | |
|---|---|---|
| True ref: | `normalizedSourcePath.Substring(0, indexOfHash)` | |
| SLM (this work) | **`normalizedSourcePath.Substring(0, indexOfHash)`** | (28.3%) |
| | `normalizedSourcePath.Substring(1)` | (8.8%) |
| | `normalizedSourcePath.Remove(indexOfHash)` | (8.2%) |
| BiLSTM→LSTM +copy | `indexOfHash + "<UNK>"` | |
| | `indexOfHash > normalizedOfHash` | |
| | `indexOfHash > 0` | |
| *GNN*→$\mathcal{NAG}$ | `normalizedSourcePath + normalizeZipFilePath` | |
| | `normalizedSourcePath + normalizedSourcePath` | |
| | `normalizedSourcePath + normalizeZipFilePath + <UNK>` | |

*Figure 21.* C# examples from our test set of the restricted completion task along with the predictions of our model and the baselines.